Bayesian Deep Learning:

Basics, Framework, and Concrete Models

Hao Wang

RUTGERS
UNIVERSITY | NEW BRUNSWICK

Perception

- See (visual object recognition)
- Read (text understanding)
- Hear (speech recognition)

Perception: perceive the environment

Inference

- Think (**inference** & **reasoning**)

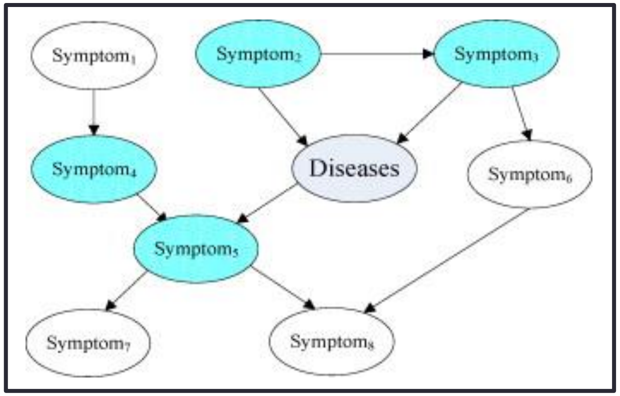
Perception

- See (visual object recognition)
- Read (text understanding)
- Hear (speech recognition)

Perception: perceive the environment

Inference

- Think (**inference** & **reasoning**)



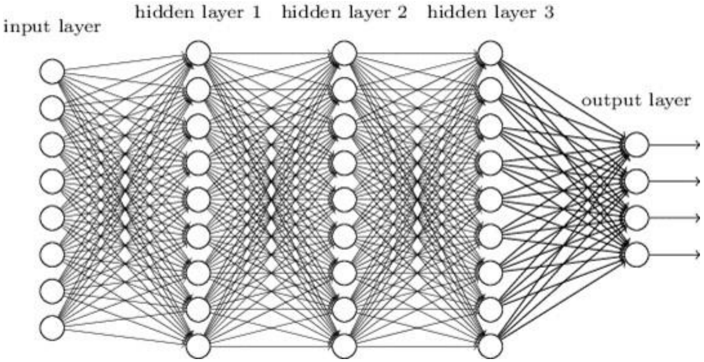
Complex relations

Conditional dependencies & randomness

Perception

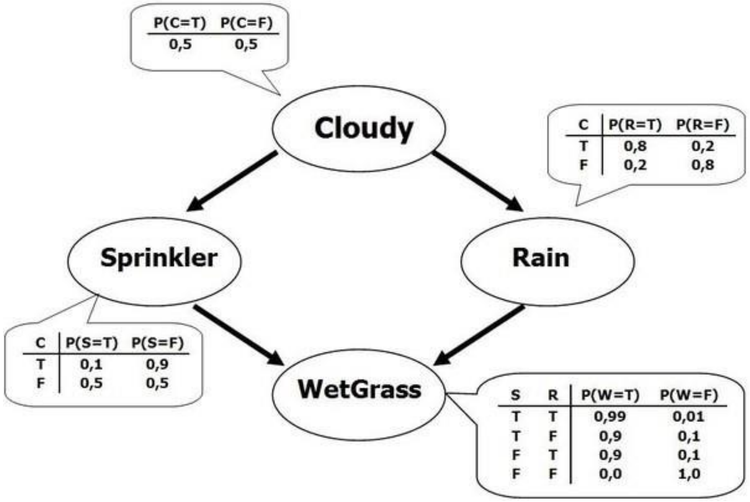
Deep Learning

High dimensional input:
Text, Images, Videos



Inference

Graphical Models

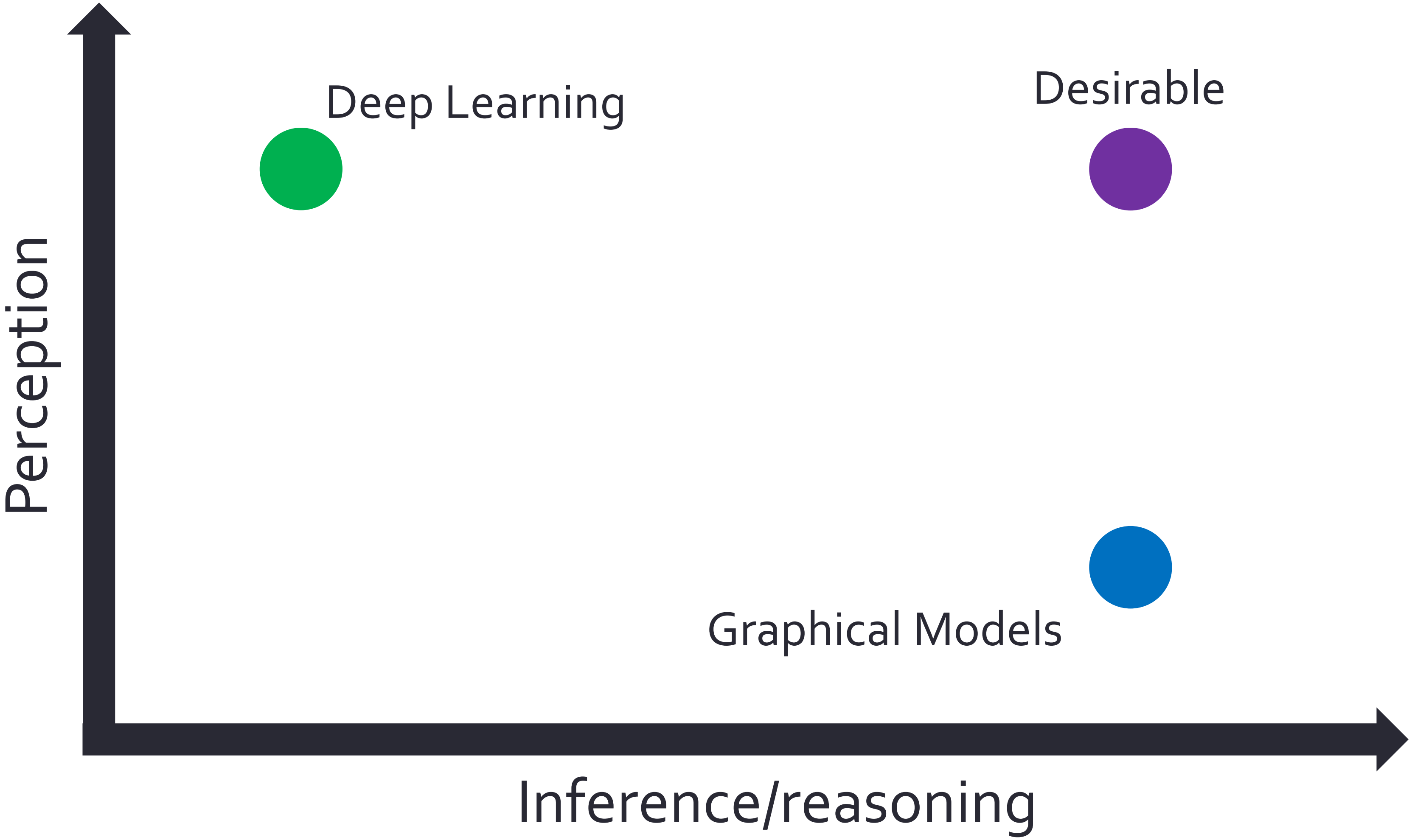


P(C=T)	P(C=F)
0,5	0,5

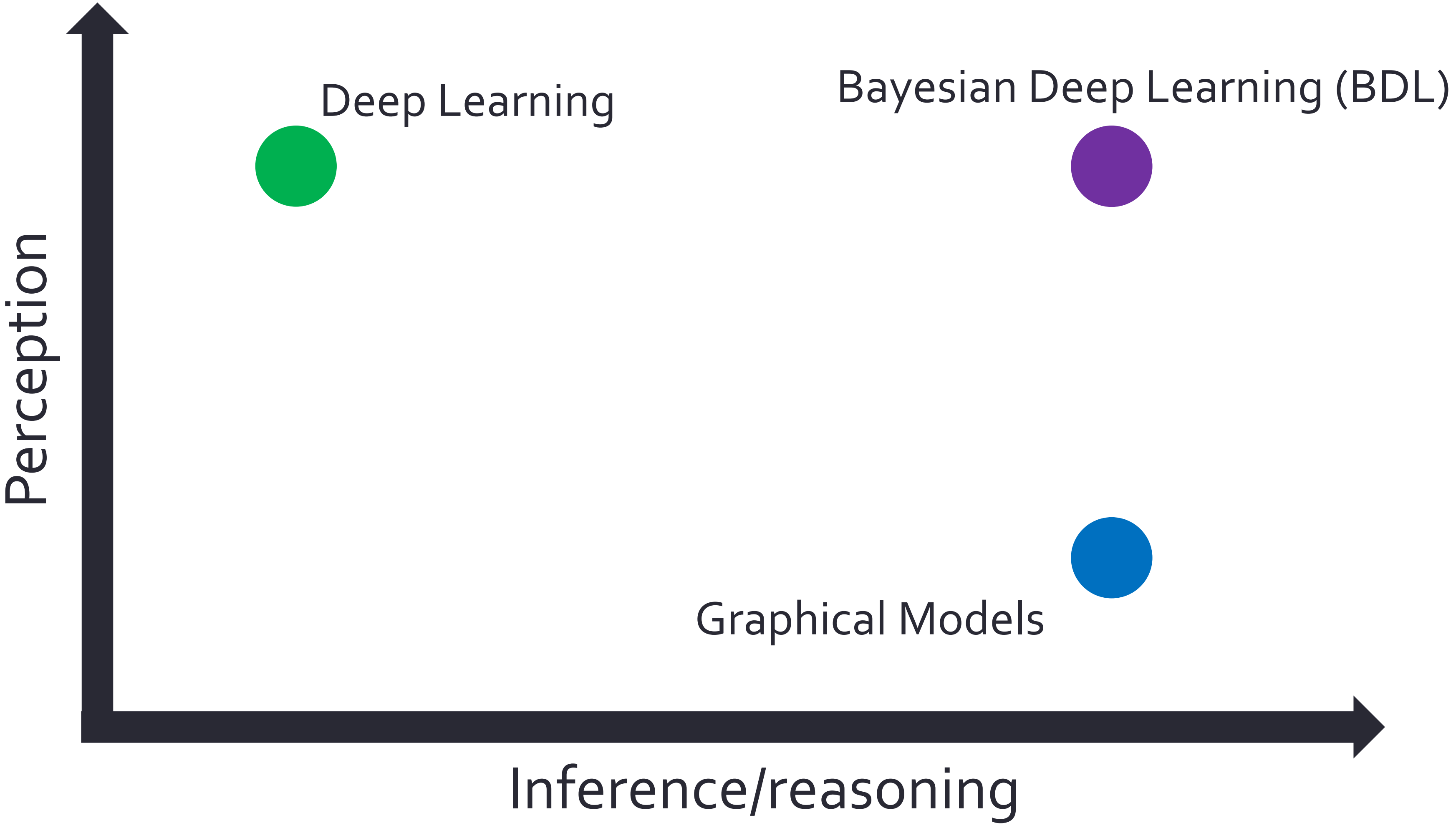
C	P(R=T)	P(R=F)
T	0,8	0,2
F	0,2	0,8

C	P(S=T)	P(S=F)
T	0,1	0,9
F	0,5	0,5

S	R	P(W=T)	P(W=F)
T	T	0,99	0,01
T	F	0,9	0,1
F	T	0,9	0,1
F	F	0,0	1,0

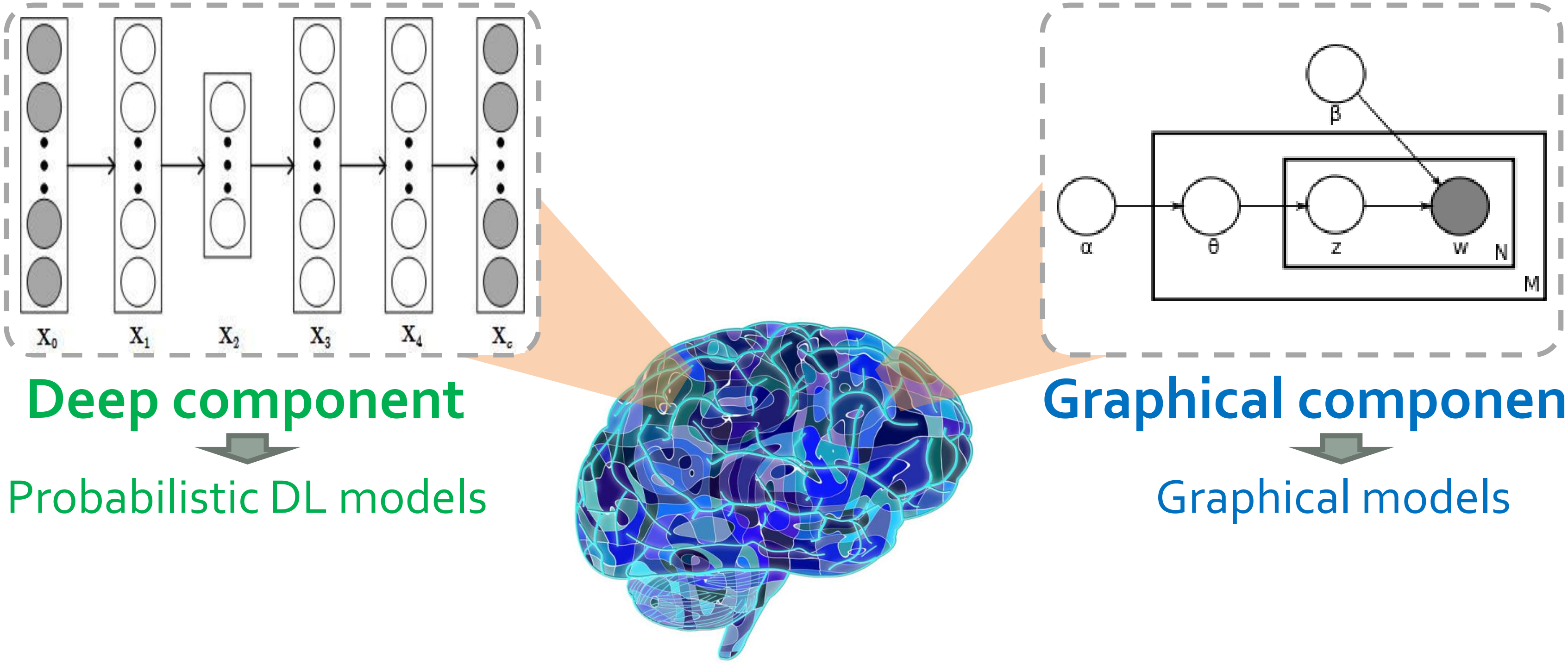


[**Wang** et al. 2014]
[**Wang** et al. 2016]
[**Wang** et al. 2020]



[**Wang** et al. 2014]
[**Wang** et al. 2016]
[**Wang** et al. 2020]

Bayesian Deep Learning (BDL)



Deep component

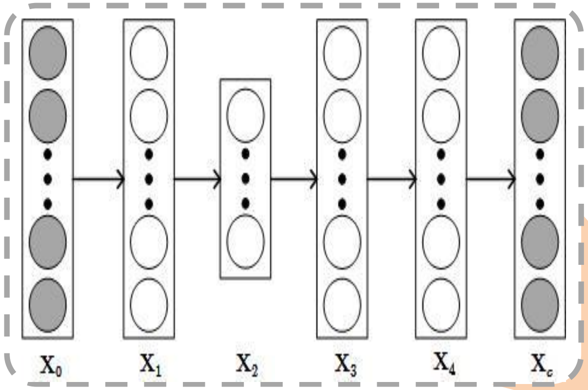
Probabilistic DL models

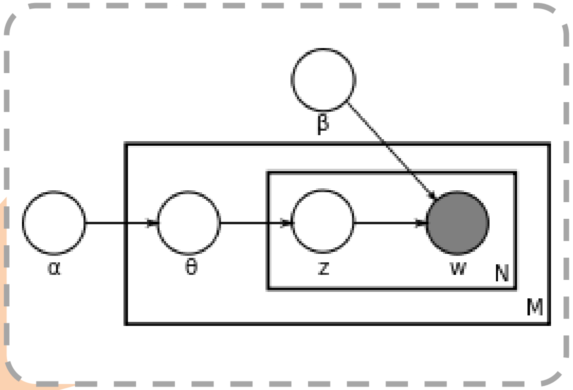
Graphical component

Graphical models

Bayesian deep learning (BDL)

Bayesian Deep Learning (BDL)



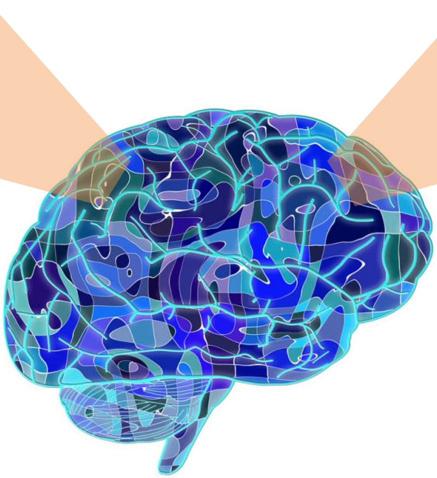


Deep component

Probabilistic DL models

Graphical component

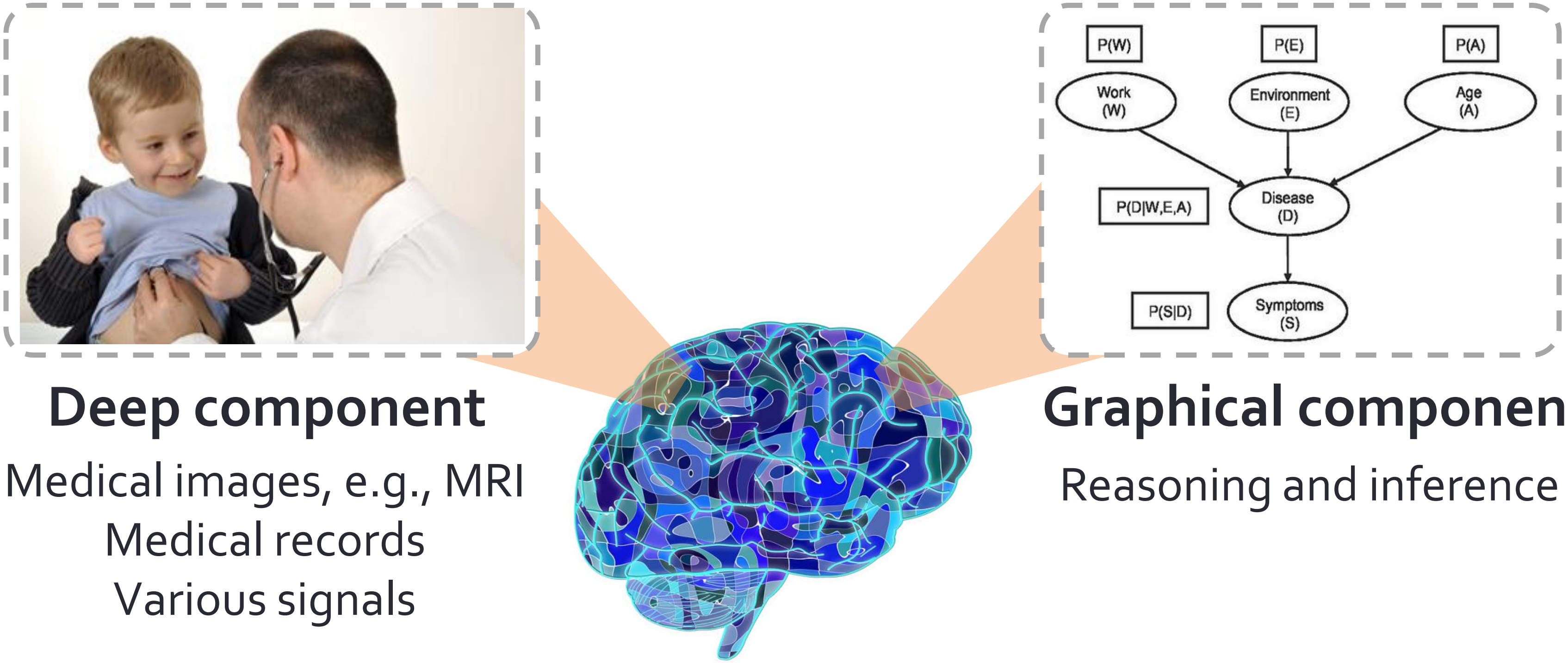
Graphical models



Bayesian deep learning (BDL)

- Maximum a posteriori (MAP)
- Markov chain Monte Carlo (MCMC)
- Variational inference (VI)

Example: Medical Diagnosis



Deep component

Medical images, e.g., MRI
Medical records
Various signals

Graphical component

Reasoning and inference

Bayesian deep learning (BDL)

[**Wang** et al., ICML 2020]
[Zhao*, Hoti*, **Wang**, Raghu, Katabi, Nature Medicine 2021]

Example: Movie Recommender Systems

Deep component

Uses video, plot, actors, etc.
Content understanding

user movie	1	2	3	4	5
1	✓	?	?	?	?
2	✓	?	?	✓	?
3	?	?	✓	?	?
4	?	✓	?	?	✓
5	✓	?	?	?	?

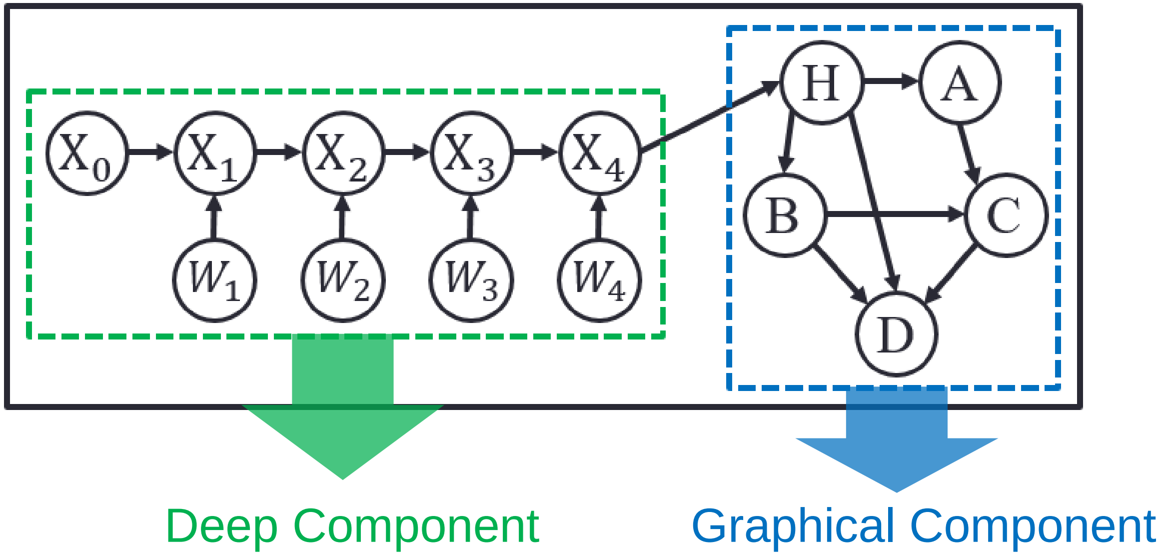
Graphical component

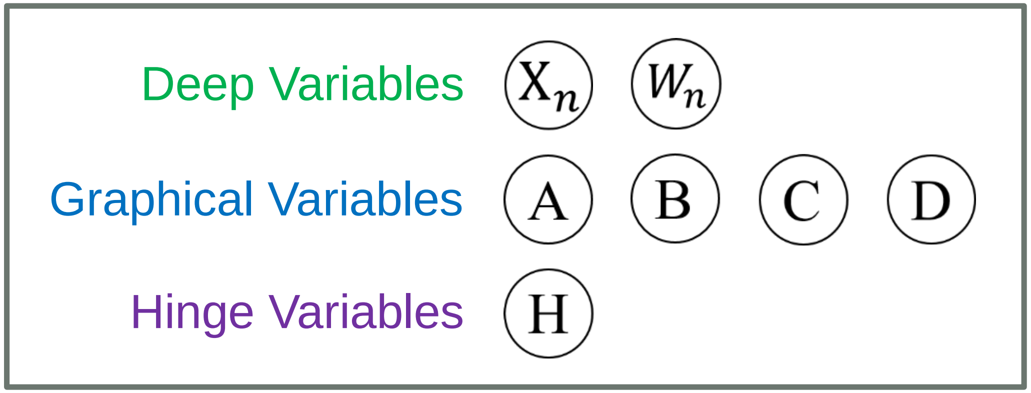
Uses preferences, similarities
Recommendation

Bayesian deep learning (BDL)

[**Wang** et al., KDD 2015]
[**Wang** et al., NIPS 2016a]

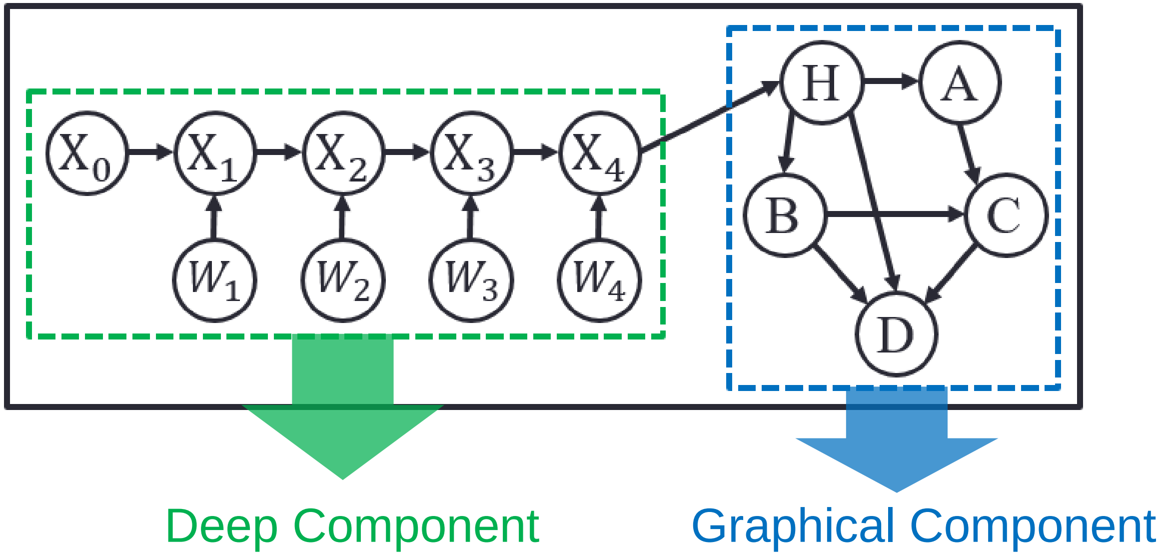
BDL: A Principled Probabilistic Framework





[**Wang** et al. TKDE 2016]

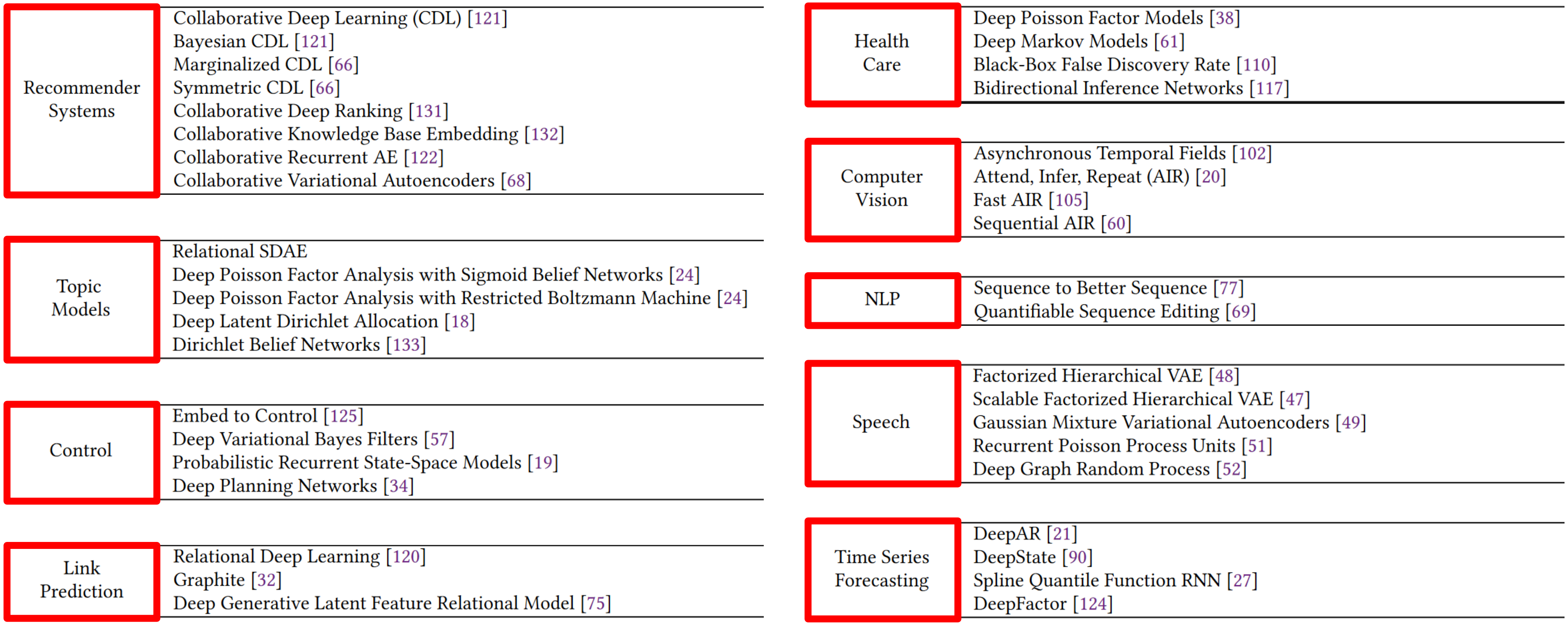
BDL: A Principled Probabilistic Framework



The whole model is **jointly** learned (**end-to-end**).

[**Wang** et al. TKDE 2016]
[**Wang** et al. CSUR 2020]

BDL Models for Different Applications



Recommender Systems	Collaborative Deep Learning (CDL) [121] Bayesian CDL [121] Marginalized CDL [66] Symmetric CDL [66] Collaborative Deep Ranking [131] Collaborative Knowledge Base Embedding [132] Collaborative Recurrent AE [122] Collaborative Variational Autoencoders [68]
Topic Models	Relational SDAE Deep Poisson Factor Analysis with Sigmoid Belief Networks [24] Deep Poisson Factor Analysis with Restricted Boltzmann Machine [24] Deep Latent Dirichlet Allocation [18] Dirichlet Belief Networks [133]
Control	Embed to Control [125] Deep Variational Bayes Filters [57] Probabilistic Recurrent State-Space Models [19] Deep Planning Networks [34]
Link Prediction	Relational Deep Learning [120] Graphite [32] Deep Generative Latent Feature Relational Model [75]
Health Care	Deep Poisson Factor Models [38] Deep Markov Models [61] Black-Box False Discovery Rate [110] Bidirectional Inference Networks [117]
Computer Vision	Asynchronous Temporal Fields [102] Attend, Infer, Repeat (AIR) [20] Fast AIR [105] Sequential AIR [60]
NLP	Sequence to Better Sequence [77] Quantifiable Sequence Editing [69]
Speech	Factorized Hierarchical VAE [48] Scalable Factorized Hierarchical VAE [47] Gaussian Mixture Variational Autoencoders [49] Recurrent Poisson Process Units [51] Deep Graph Random Process [52]
Time Series Forecasting	DeepAR [21] DeepState [90] Spline Quantile Function RNN [27] DeepFactor [124]

https://github.com/js05212/BayesianDeepLearning-Survey

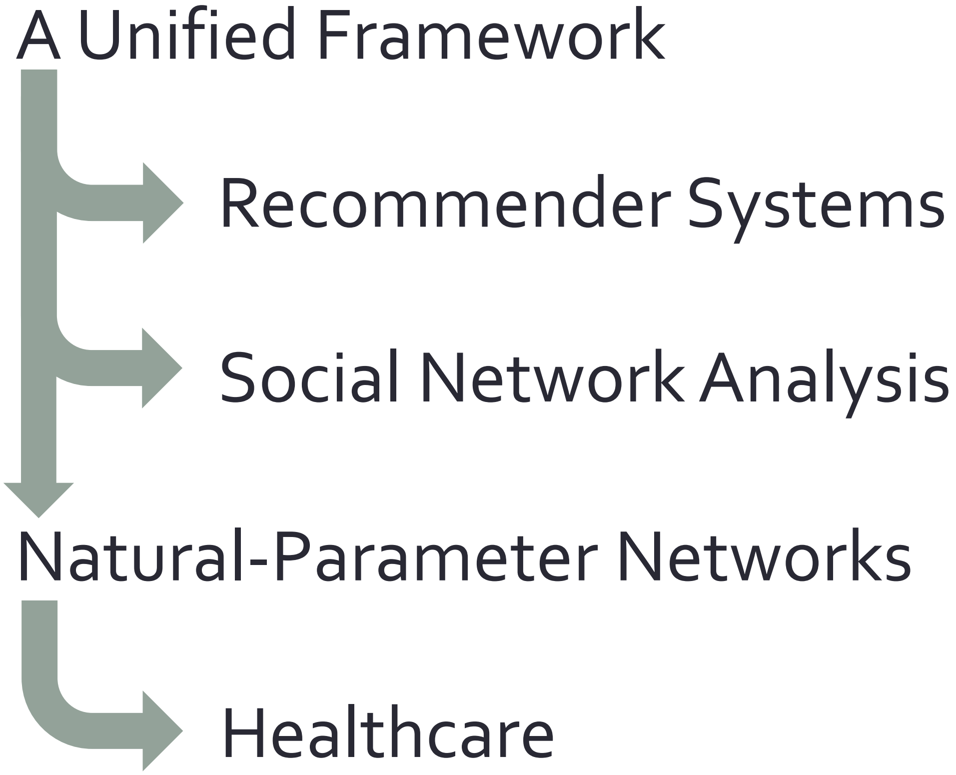
[**Wang** et al. TKDE 2016]
[**Wang** et al. CSUR 2020]

Bayesian Deep Learning

A Unified Framework

- Probabilistic Graphical Models
- Probabilistic/Bayesian Neural Nets

Bayesian Deep Learning



A Unified Framework

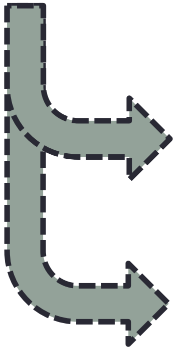
Recommender Systems

Social Network Analysis

Natural-Parameter Networks

Healthcare

Bayesian Deep Learning



A Unified Framework

Probabilistic Graphical Models

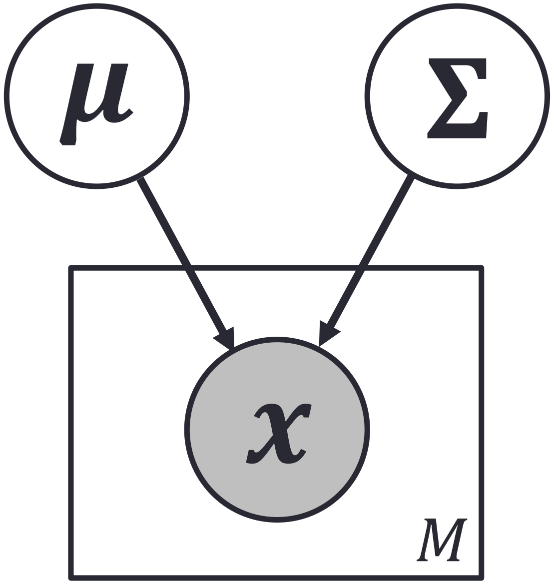
Probabilistic/Bayesian Neural Nets

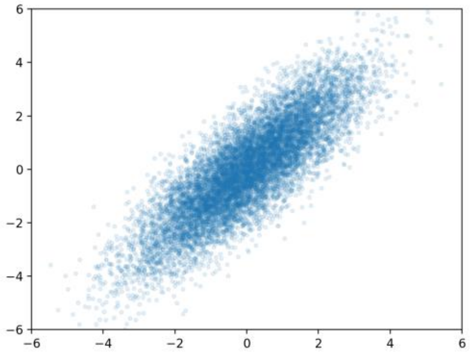
Probabilistic Graphical Models:
A Mini-Tutorial

Probabilistic Graphical Models: Simple Example

Gaussian Distribution: $\boldsymbol{x} \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$

$$\boldsymbol{x} \in R^D$$
$$\boldsymbol{\mu} \in R^D$$
$$\boldsymbol{\Sigma} \in R^{D \times D}$$





● Observed variables (**given**)

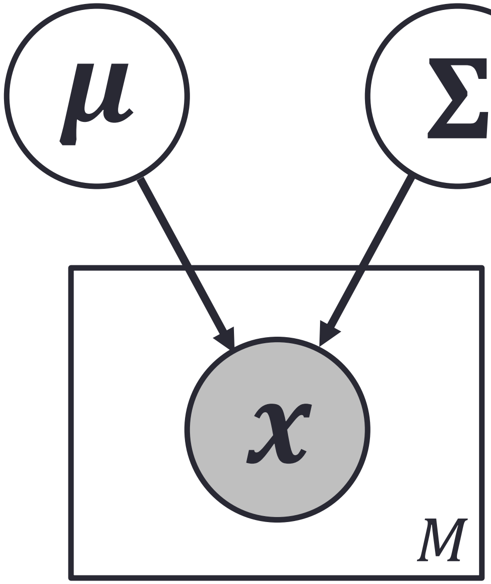
○ Latent variables & parameters **to infer/learn**

M Number of repetitions (Number of data points)

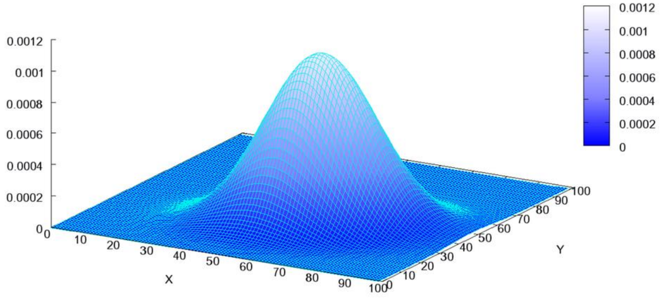
Probabilistic Graphical Models: Simple Example

Gaussian Distribution: $\boldsymbol{x} \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$

$$\boldsymbol{x} \in R^D$$
$$\boldsymbol{\mu} \in R^D$$
$$\boldsymbol{\Sigma} \in R^{D \times D}$$

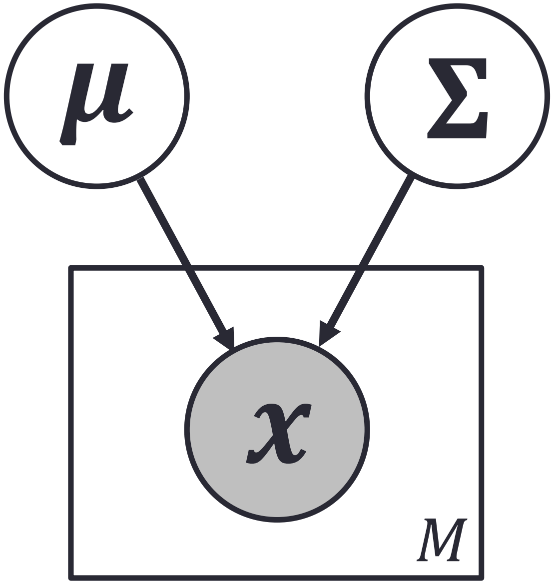


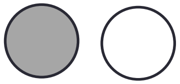
$$f_{\mathbf{X}}(x_1, \ldots, x_k) = \frac{\exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} (\mathbf{x}-\boldsymbol{\mu})\right)}{\sqrt{(2\pi)^k |\boldsymbol{\Sigma}|}}$$



Probabilistic Graphical Models: Nodes and Edges

Gaussian Distribution: $\boldsymbol{x} \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$



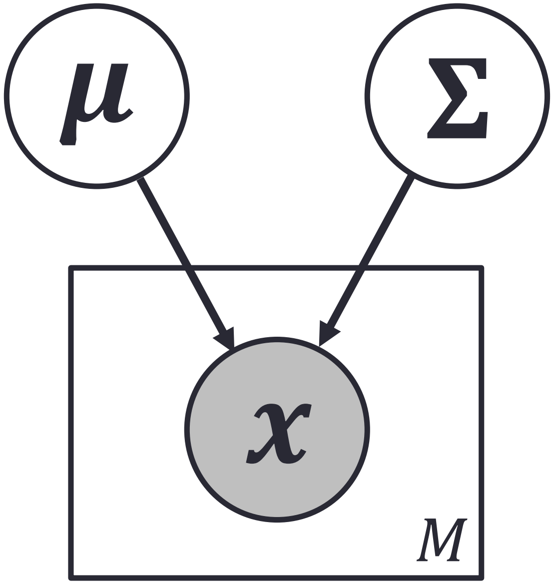


Variables (either **observed** or **latent**) or parameters : $\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_M$

Conditional dependency: $p(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = N(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$

Probabilistic Graphical Models: Generative Process

Gaussian Distribution: $\boldsymbol{x} \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$

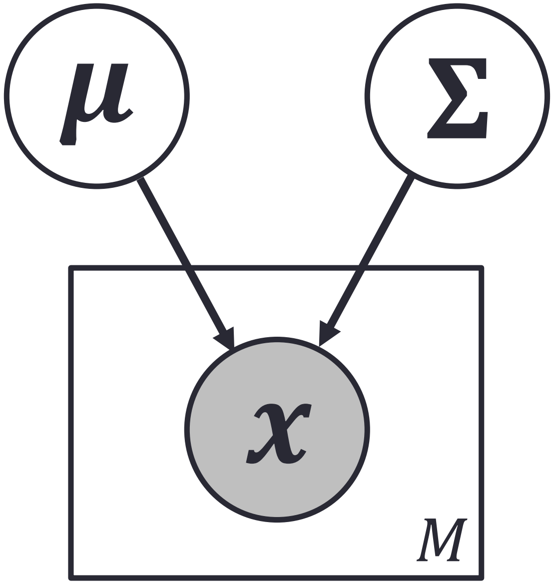


Generative process

For each $m = 1,2, \ldots, M$:
    Draw $\boldsymbol{x_m} \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$

Learning and Inference

Gaussian Distribution: $\boldsymbol{x} \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$



Learning: Given observed data, learn the unknown parameters.

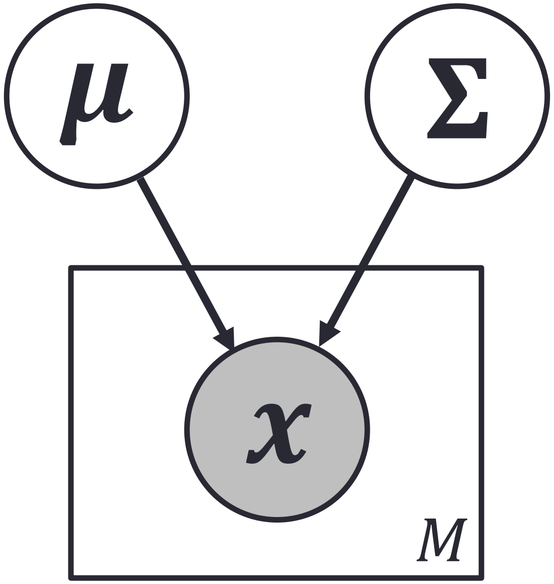
$\boldsymbol{x}\ (or\ \boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_M)$ $\boldsymbol{\mu}, \boldsymbol{\Sigma}$

Inference: Given observed data and parameters, infer the latent variables.

Not applicable in this simple example since we do not have latent variables.

Learning ~~and Inference~~

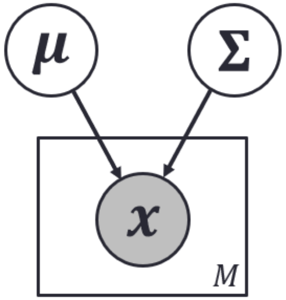
Gaussian Distribution: $\boldsymbol{x} \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$



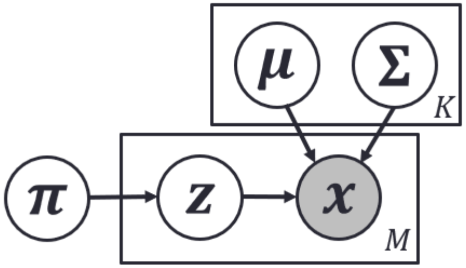
Learning: Given observed data $\boldsymbol{x}$, learn the unknown parameters $\boldsymbol{\mu}, \boldsymbol{\Sigma}$.

$$\boldsymbol{\mu} = \frac{1}{M}\sum_{m=1}^{M} \boldsymbol{x}_m , \qquad \boldsymbol{\Sigma} = \frac{1}{M}\sum_{m=1}^{M} (\boldsymbol{x}_m - \boldsymbol{\mu})(\boldsymbol{x}_m - \boldsymbol{\mu})^\top$$

Summary on Probabilistic Graphical Models



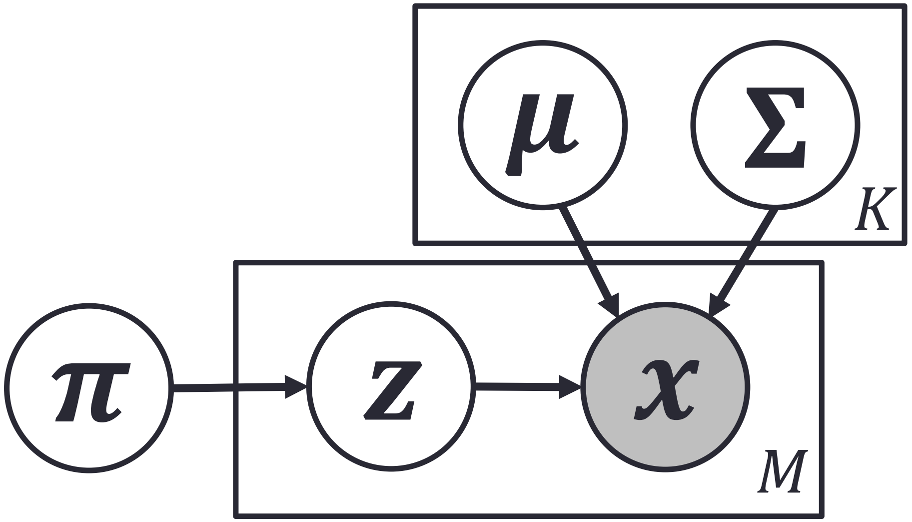
Gaussian Distribution



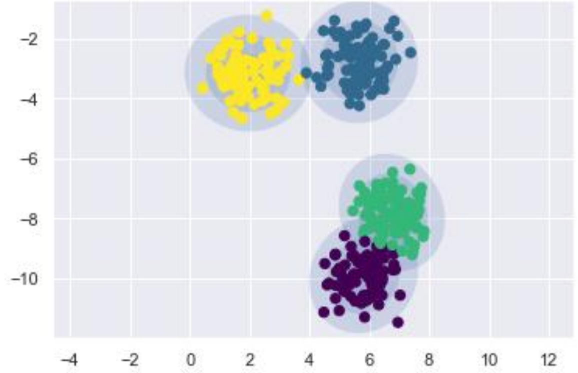
Mixture of Gaussians

Probabilistic Graphical Models: A Slightly More Complicated Example

Mixture of Gaussians

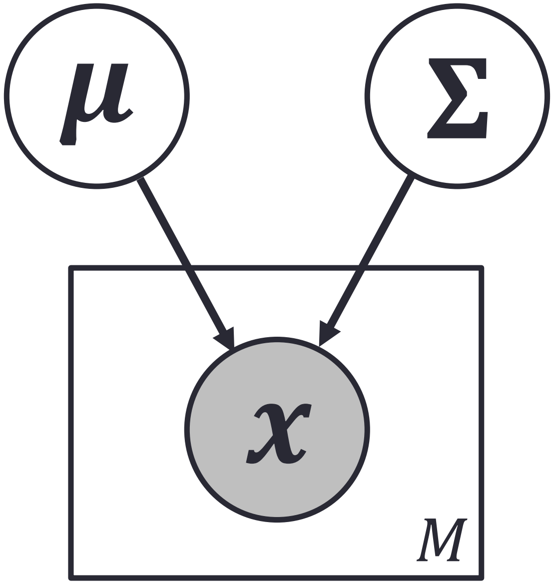


Example: Mixture of 4 Gaussians (K=4)



Generative Process for the Gaussian Model (Recap)

Gaussian Distribution: $\boldsymbol{x} \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$

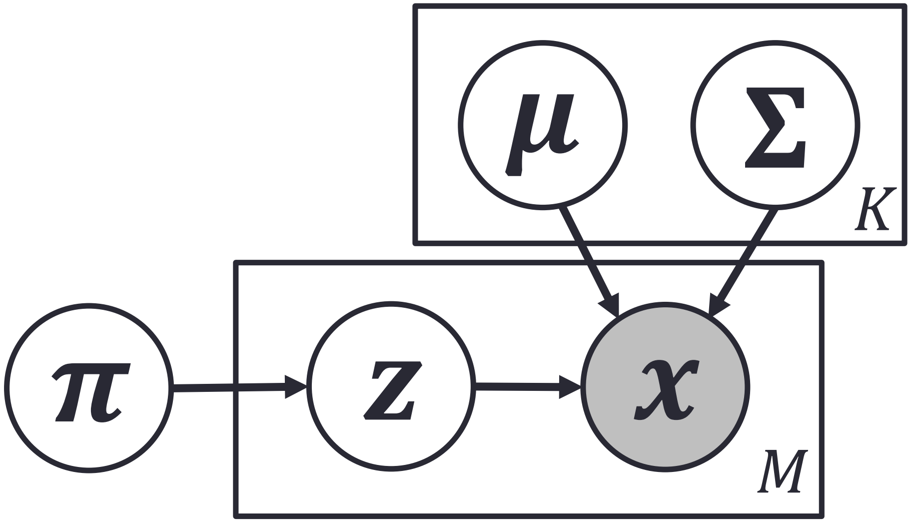


Generative process (of M data points)

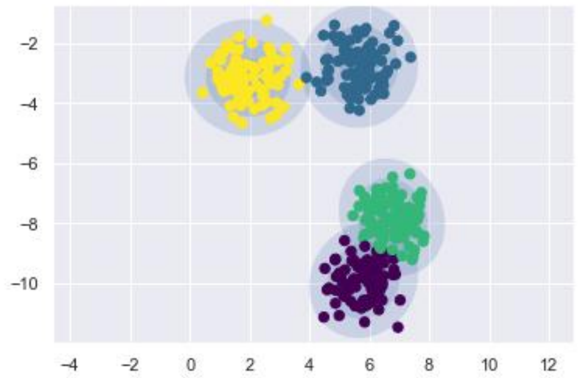
For each $m = 1,2, \dots, M$:
Draw $\boldsymbol{x_m} \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$

Mixture of Gaussians: Generative Process

Mixture of Gaussians



Example: Mixture of 4 Gaussians (K=4)



Generative process (of M data points)

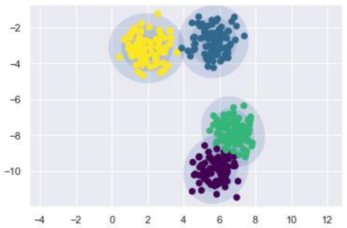
For each $m = 1,2, \ldots, M$:

Choose 1 of the K Gaussians: Draw $\boldsymbol{z_m} \sim Categorical(\boldsymbol{\pi})$

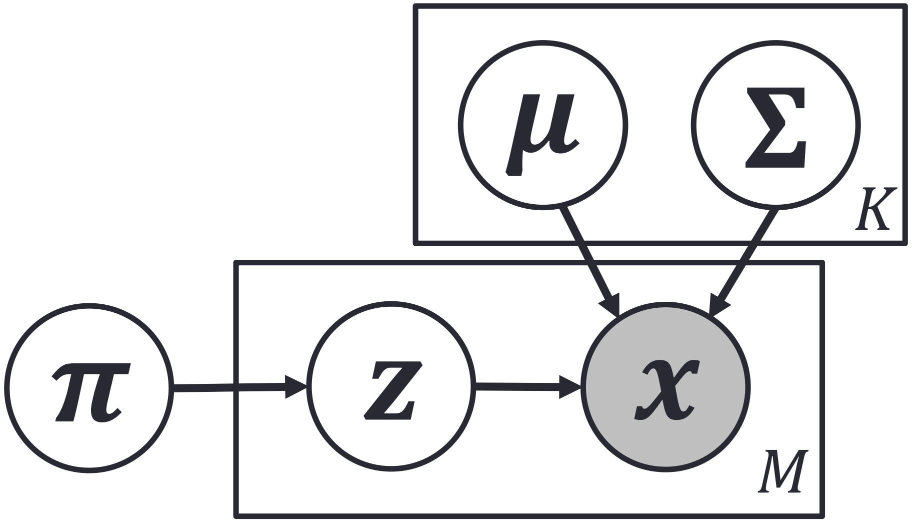
Sample from the chosen Gaussian: $\boldsymbol{x_m} \sim N(\boldsymbol{\mu_k}, \boldsymbol{\Sigma_k})$

$\pi = [0.25, 0.25, 0.25, 0.25]$, $\pi = [1.0, 0, 0, 0]$, $\pi = [0.8, 0.2, 0, 0]$

Mixture of Gaussians: Generative Process



Mixture of Gaussians



Real-value K-dim vector:

$$\boldsymbol{\pi} = [\boldsymbol{\pi}^{(1)}, \dots, \boldsymbol{\pi}^{(k)}, \dots, \boldsymbol{\pi}^{(K)}]$$

$$0 \le \boldsymbol{\pi}^{(k)} \le 1, \sum_{k=1}^{K} \boldsymbol{\pi}^{(k)} = 1$$

One-hot K-dim vector:

$$\boldsymbol{z_m} = [\boldsymbol{z_m}^{(1)}, \dots, \boldsymbol{z_m}^{(k)}, \dots, \boldsymbol{z_m}^{(K)}]$$

$$\boldsymbol{z_m}^{(k)} \in \{0,1\}, \sum_{k=1}^{K} \boldsymbol{z_m}^{(k)} = 1$$

Parameters for K gaussians:

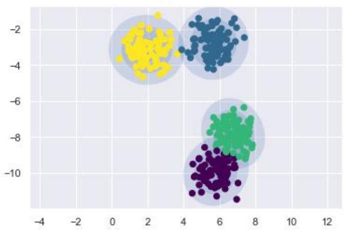
$$\boldsymbol{\mu_k}, \boldsymbol{\Sigma_k} \ (k = 1,2, \dots, K)$$

Generative process

For each $m = 1,2, \dots, M$:

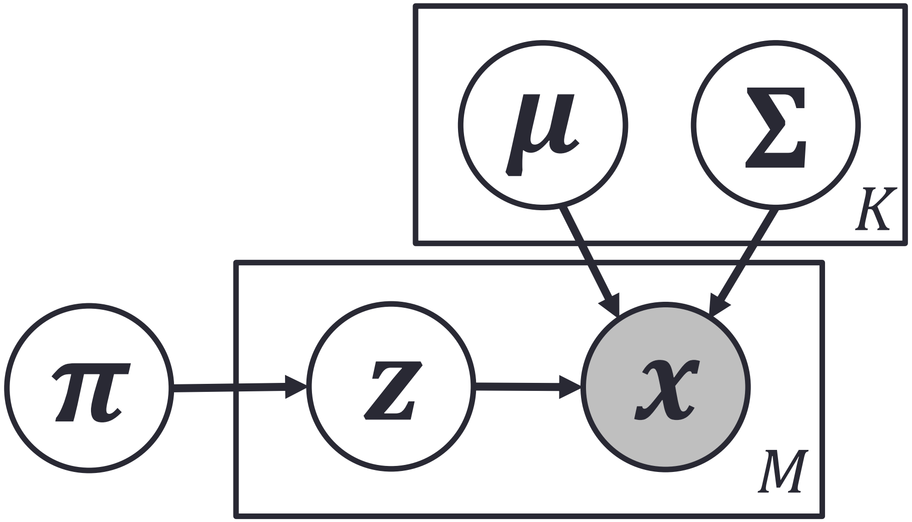
Choose 1 of the K Gaussians: Draw $\boldsymbol{z_m} \sim Categorical(\boldsymbol{\pi})$

Sample from the chosen Gaussian: $\boldsymbol{x_m} \sim N(\boldsymbol{\mu_k}, \boldsymbol{\Sigma_k})$



Mixture of Gaussians: Factorization

Mixture of Gaussians



Joint distribution expressed as:

$$p(\boldsymbol{x}_m, \boldsymbol{z}_m) = p(\boldsymbol{z}_m)p(\boldsymbol{x}_m|\boldsymbol{z}_m)$$

Choose 1 of the K Gaussians:

$$p(\boldsymbol{z}_m) = \prod_{k=1}^{K} (\boldsymbol{\pi}^{(k)})^{z_m^{(k)}}$$

Sample from the chosen Gaussian (k-th):

$$p\left(\boldsymbol{x}_m \middle| z_m^{(k)} = 1\right) = N(\boldsymbol{x}_m|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

$$p(\boldsymbol{x}_m|\boldsymbol{z}_m) = \prod_{k=1}^{K} N(\boldsymbol{x}_m|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)^{z_m^{(k)}}$$

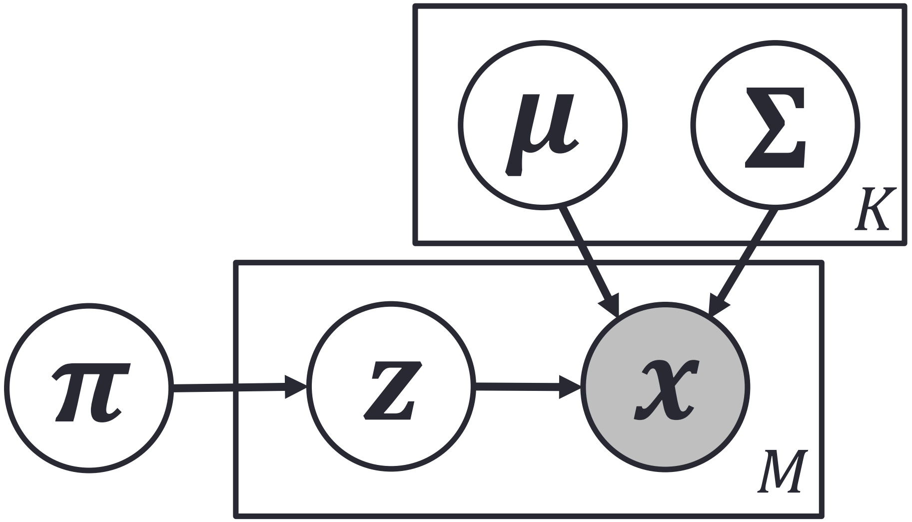
Generative process

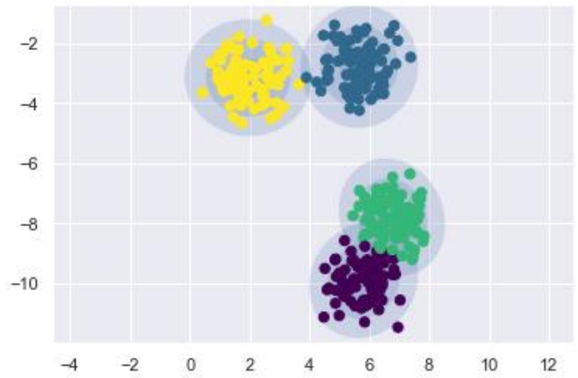
For each $m = 1,2, \dots, M$:

Choose 1 of the K Gaussians: Draw $\boldsymbol{z}_m \sim Categorical(\boldsymbol{\pi})$

Sample from the chosen Gaussian: $\boldsymbol{x}_m \sim N(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_\mathbf{k})$

Mixture of Gaussians : Nodes and Edges


Mixture of Gaussians


Example: Mixture of 4 Gaussians (K=4)

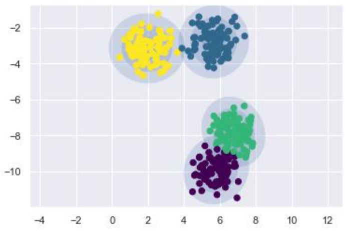
● ○ Variables (either **observed** or **latent**) or parameters :

$x_1, x_2, \ldots, x_M \quad z_1, z_2, \ldots, z_M \quad \mu_k, \Sigma_k,$

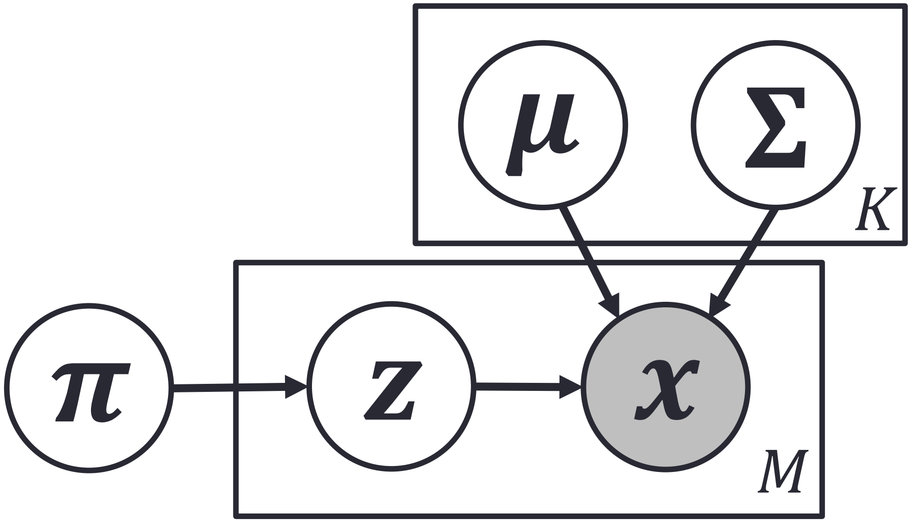
→ Conditional dependency:

$p(z_m|\pi)$ *and* $p(x_m|z_m, \pi, \{\mu_k, \Sigma_k\})$

Mixture of Gaussians : Learning and Inference



Mixture of Gaussians



Real-value K-dim vector:

$$\boldsymbol{\pi} = [\boldsymbol{\pi}^{(1)}, \dots, \boldsymbol{\pi}^{(k)}, \dots, \boldsymbol{\pi}^{(K)}]$$

$$0 \leq \boldsymbol{\pi}^{(k)} \leq 1, \sum_{k=1}^{K} \boldsymbol{\pi}^{(k)} = 1$$

One-hot K-dim vector:

$$\boldsymbol{z_m} = [\boldsymbol{z_m^{(1)}}, \dots, \boldsymbol{z_m^{(k)}}, \dots, \boldsymbol{z_m^{(K)}}]$$

$$\boldsymbol{z_m^{(k)}} \in \{0,1\}, \sum_{k=1}^{K} \boldsymbol{z_m^{(k)}} = 1$$

Parameters for K gaussians:

$\boldsymbol{\mu_k}, \boldsymbol{\Sigma_k}\ (k = 1,2, \dots, K)$

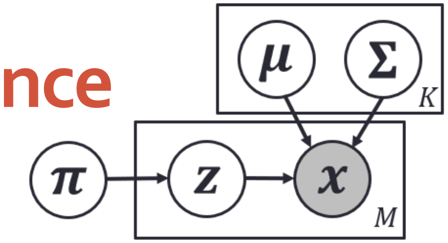
Learning: Given observed data, learn the unknown parameters.

$\boldsymbol{x_m}\ (or\ \boldsymbol{x_1}, \boldsymbol{x_2}, \dots, \boldsymbol{x_M})$ $\boldsymbol{\pi}, \boldsymbol{\mu_k}, \boldsymbol{\Sigma_k}$

Inference: Given observed data and parameters, infer the latent variables.

$\boldsymbol{x_m}$ $\boldsymbol{\pi}, \boldsymbol{\mu_k}, \boldsymbol{\Sigma_k}$ $\boldsymbol{z_m}$

Mixture of Gaussians: Learning and Inference using Expectation-Maximization (EM)



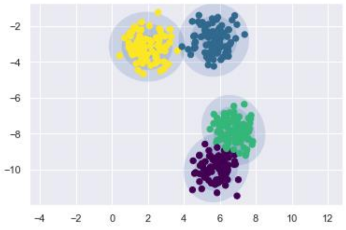
Learning: Given observed data $\boldsymbol{x}_m$, learn the parameters $\boldsymbol{\pi}, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_\mathbf{k}$

1. **Initialize** the means $\boldsymbol{\mu}_k$, covariances $\boldsymbol{\Sigma}_\mathbf{k}$ and mixing coefficients $\boldsymbol{\pi}$.

2. **E Step.** Infer the expectation (distribution) of $\boldsymbol{z}_m$, denoted as $\gamma\left(\boldsymbol{z}_m^{(k)}\right)$, given the current parameters $\boldsymbol{\pi}$, $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_\mathbf{k}$.

3. **M Step.** Update the parameters $\boldsymbol{\pi}, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_\mathbf{k}$ given the current $\gamma\left(\boldsymbol{z}_m^{(k)}\right)$.

4. Iterate between **E step** and **M step** until convergence.



Inference and Learning: E Step

$$f_{\mathbf{X}}(x_1,\ldots,x_k)=\frac{\exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)}{\sqrt{(2\pi)^k|\boldsymbol{\Sigma}|}}$$

Learning: Given observed data $\boldsymbol{x}_{\boldsymbol{m}}$, learn the parameters $\boldsymbol{\pi}, \boldsymbol{\mu}_{\boldsymbol{k}}, \boldsymbol{\Sigma}_{\mathbf{k}}$

1. **Initialize** the means $\boldsymbol{\mu}_{\boldsymbol{k}}$, covariances $\boldsymbol{\Sigma}_{\mathbf{k}}$ and mixing coefficients $\boldsymbol{\pi}$.

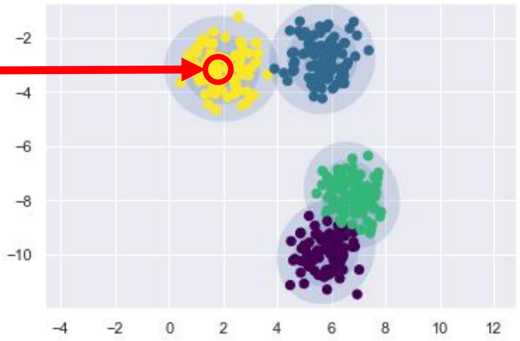
2. **E Step.** Infer the expectation (distribution) of $\boldsymbol{z}_{\boldsymbol{m}}$ given the current parameters.

$$\gamma\left(\boldsymbol{z}_{\boldsymbol{m}}^{(\boldsymbol{k})}\right)=p\left(\boldsymbol{z}_{\boldsymbol{m}}^{(\boldsymbol{k})}=1\middle|\boldsymbol{x}_{\boldsymbol{m}},\boldsymbol{\pi},\{\boldsymbol{\mu}_{\boldsymbol{k}},\boldsymbol{\Sigma}_{\boldsymbol{k}}\}\right)$$

$$\propto p\left(\boldsymbol{z}_{\boldsymbol{m}}^{(\boldsymbol{k})}=1\right)p\left(\boldsymbol{x}_{\boldsymbol{m}}\middle|\boldsymbol{z}_{\boldsymbol{m}}^{(\boldsymbol{k})}=1\right)=\boldsymbol{\pi}^{(\boldsymbol{k})}N(\boldsymbol{x}_{\boldsymbol{m}}|\boldsymbol{\mu}_{\boldsymbol{k}},\boldsymbol{\Sigma}_{\mathbf{k}})$$

$$\sum_{k=1}^{K}\gamma\left(\boldsymbol{z}_{\boldsymbol{m}}^{(\boldsymbol{k})}\right)=1$$

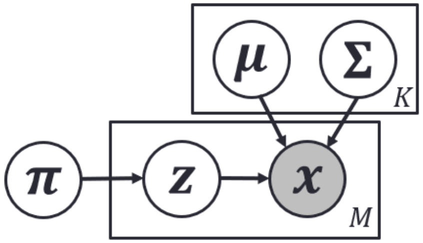
E Step tries to infer the probability that this point $\boldsymbol{x}_{\boldsymbol{1}}$ belongs to each Gaussian.



$\gamma\left(\boldsymbol{z}_{\boldsymbol{1}}^{(\boldsymbol{1})}\right)$ is large.

$\gamma\left(\boldsymbol{z}_{\boldsymbol{1}}^{(\boldsymbol{2})}\right), \gamma\left(\boldsymbol{z}_{\boldsymbol{1}}^{(\boldsymbol{3})}\right), \gamma\left(\boldsymbol{z}_{\boldsymbol{1}}^{(\boldsymbol{4})}\right)$ are small.

Inference and Learning: E Step



Learning: Given observed data $\boldsymbol{x_m}$, learn the parameters $\boldsymbol{\pi}, \boldsymbol{\mu_k}, \boldsymbol{\Sigma_{\mathrm{k}}}$

1. **Initialize** the means $\boldsymbol{\mu_k}$, covariances $\boldsymbol{\Sigma_{\mathrm{k}}}$ and mixing coefficients $\boldsymbol{\pi}$.

2. **E Step.** Infer the expectation (distribution) of $\boldsymbol{z_m}$ given the current parameters.

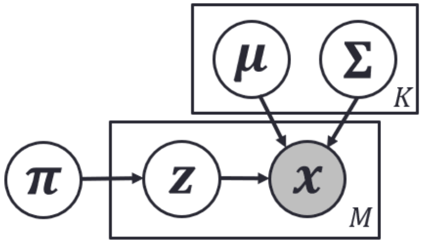
$$\gamma\left(\boldsymbol{z_m^{(k)}}\right) = p\left(\boldsymbol{z_m^{(k)}} = 1 \middle| \boldsymbol{x_m}, \boldsymbol{\pi}, \{\boldsymbol{\mu_k}, \boldsymbol{\Sigma_k}\}\right)$$

$$= \frac{p\left(\boldsymbol{z_m^{(k)}} = 1\right) p\left(\boldsymbol{x_m} \middle| \boldsymbol{z_m^{(k)}} = 1\right)}{\sum_{j=1}^{K} p\left(\boldsymbol{z_m^{(j)}} = 1\right) p\left(\boldsymbol{x_m} \middle| \boldsymbol{z_m^{(j)}} = 1\right)} = \frac{\boldsymbol{\pi}^{(k)} N(\boldsymbol{x_m} | \boldsymbol{\mu_k}, \boldsymbol{\Sigma_{\mathrm{k}}})}{\sum_{j=1}^{K} \boldsymbol{\pi}^{(j)} N(\boldsymbol{x_m} | \boldsymbol{\mu_j}, \boldsymbol{\Sigma_j})}$$

Bayes' Rule:

$$p(z|x) = \frac{p(x,z)}{p(x)} = \frac{p(z)p(x|z)}{\sum_z p(x,z)} = \frac{p(z)p(x|z)}{\sum_z p(z)p(x|z)}$$

Inference and Learning: M Step



Learning: Given observed data $\boldsymbol{x_m}$, learn the parameters $\boldsymbol{\pi}, \boldsymbol{\mu_k}, \boldsymbol{\Sigma_k}$

1. **Initialize** the means $\boldsymbol{\mu_k}$, covariances $\boldsymbol{\Sigma_k}$ and mixing coefficients $\boldsymbol{\pi}$.
2. **E Step.** Infer the expectation (distribution) of $\boldsymbol{z_m}$, denoted as $\gamma\left(\boldsymbol{z_m^{(k)}}\right)$, given the current parameters $\boldsymbol{\pi}$, $\boldsymbol{\mu_k}$ and $\boldsymbol{\Sigma_k}$.
3. **M Step.** Update the parameters $\boldsymbol{\pi}, \boldsymbol{\mu_k}, \boldsymbol{\Sigma_k}$ given the current $\gamma\left(\boldsymbol{z_m^{(k)}}\right)$.

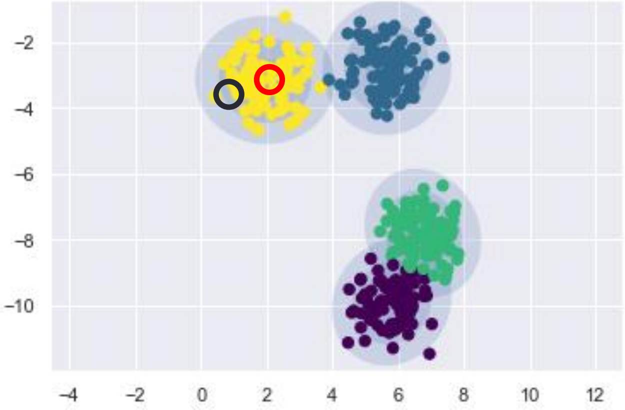
$$\boldsymbol{\mu_k} = \frac{1}{M_k}\sum_{m=1}^{M} \gamma(\boldsymbol{z_m^{(k)}})\boldsymbol{x_m}$$

where $M_k = \sum_{m=1}^{M} \gamma(\boldsymbol{z_m^{(k)}})$

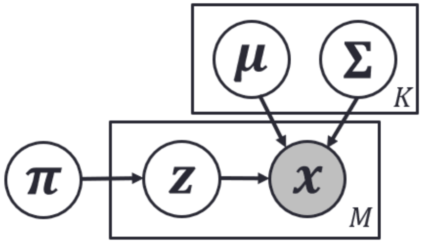
Intuition for updating $\boldsymbol{\mu_k}$:

(a) Gather data points $\boldsymbol{x_m}$ which are assigned to the same Gaussian, and compute their average

(b) Data points that belong to the Gaussian 'more' will have larger weight $\gamma\left(\boldsymbol{z_m^{(k)}}\right)$



Inference and Learning: M Step



Learning: Given observed data $\boldsymbol{x_m}$, learn the parameters $\boldsymbol{\pi}, \boldsymbol{\mu_k}, \boldsymbol{\Sigma_\mathrm{k}}$

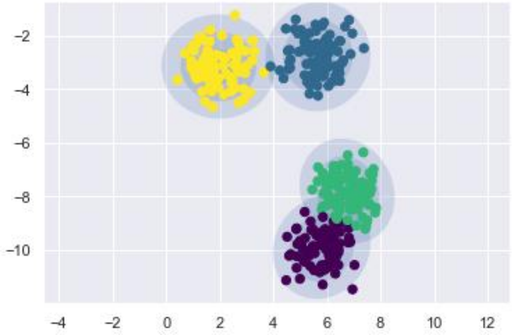
1. **Initialize** the means $\boldsymbol{\mu_k}$, covariances $\boldsymbol{\Sigma_\mathrm{k}}$ and mixing coefficients $\boldsymbol{\pi}$.
2. **E Step.** Infer the expectation (distribution) of $\boldsymbol{z_m}$, denoted as $\gamma\left(\boldsymbol{z_m^{(k)}}\right)$, given the current parameters $\boldsymbol{\pi}$, $\boldsymbol{\mu_k}$ and $\boldsymbol{\Sigma_\mathrm{k}}$.
3. **M Step.** Update the parameters $\boldsymbol{\pi}, \boldsymbol{\mu_k}, \boldsymbol{\Sigma_\mathrm{k}}$ given the current $\gamma\left(\boldsymbol{z_m^{(k)}}\right)$.

$$\boldsymbol{\mu_k} = \frac{1}{M_k} \sum_{m=1}^{M} \gamma(\boldsymbol{z_m^{(k)}}) \boldsymbol{x_m}$$

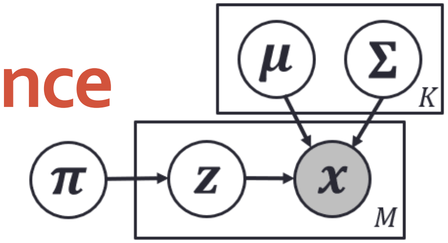
$$\boldsymbol{\Sigma_k} = \frac{1}{M_k} \sum_{m=1}^{M} \gamma(\boldsymbol{z_m^{(k)}}) (\boldsymbol{x_m} - \boldsymbol{\mu_k})(\boldsymbol{x_m} - \boldsymbol{\mu_k})^\top$$

$$\boldsymbol{\pi}^{(\boldsymbol{k})} = \frac{M_k}{M}$$

where $M_k = \sum_{m=1}^{M} \gamma(\boldsymbol{z_m^{(k)}})$



Mixture of Gaussians: Learning and Inference using Expectation-Maximization (EM)

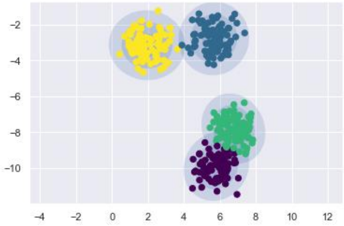


Learning: Given observed data x_m , learn the parameters $\pi, \mu_k, \Sigma_\mathrm{k}$

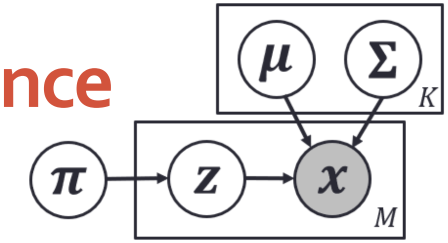
1. **Initialize** the means μ_k, covariances Σ_k and mixing coefficients π.
2. **E Step.** Infer the expectation (distribution) of z_m, denoted as $\gamma\left(z_m^{(k)}\right)$, given the current parameters π, μ_k and Σ_k.
3. **M Step.** Update the parameters $\pi, \mu_k, \Sigma_\mathrm{k}$ given the current $\gamma\left(z_m^{(k)}\right)$.
4. Iterate between **E step** and **M step** until <u>convergence</u>.

One last problem:

What convergence criterion to use?



Mixture of Gaussians: Learning and Inference using Expectation-Maximization (EM)

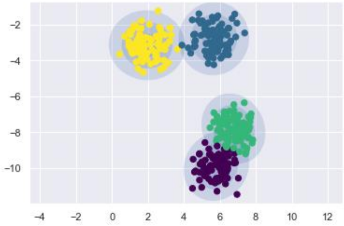


Learning: Given observed data $\boldsymbol{x_m}$, learn the parameters $\boldsymbol{\pi}, \boldsymbol{\mu_k}, \boldsymbol{\Sigma_\mathrm{k}}$

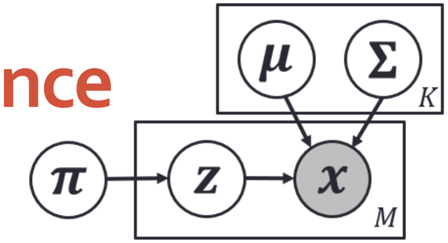
1. **Initialize** the means $\boldsymbol{\mu_k}$, covariances $\boldsymbol{\Sigma_\mathrm{k}}$ and mixing coefficients $\boldsymbol{\pi}$.
2. **E Step.** Infer the expectation (distribution) of $\boldsymbol{z_m}$, denoted as $\gamma\left(\boldsymbol{z_m^{(k)}}\right)$, given the current parameters $\boldsymbol{\pi}$, $\boldsymbol{\mu_k}$ and $\boldsymbol{\Sigma_\mathrm{k}}$.
3. **M Step.** Update the parameters $\boldsymbol{\pi}, \boldsymbol{\mu_k}, \boldsymbol{\Sigma_\mathrm{k}}$ given the current $\gamma\left(\boldsymbol{z_m^{(k)}}\right)$.
4. Iterate between **E step** and **M step** until convergence.

Likelihood for $\boldsymbol{x_m}$: $p(\boldsymbol{x_m}) = \sum_{\boldsymbol{z_m}} p(\boldsymbol{z_m}) p(\boldsymbol{x_m}|\boldsymbol{z_m}) = \sum_{k=1}^{K} \boldsymbol{\pi}^{(\boldsymbol{k})}\, N(\boldsymbol{x_m}|\boldsymbol{\mu_k}, \boldsymbol{\Sigma_\mathrm{k}})$

Log-likelihood for M data points: $\mathrm{L} = \sum_{m=1}^{\mathrm{M}} \log[\sum_{k=1}^{K} \boldsymbol{\pi}^{(\boldsymbol{k})}\, N(\boldsymbol{x_m}|\boldsymbol{\mu_k}, \boldsymbol{\Sigma_\mathrm{k}})]$

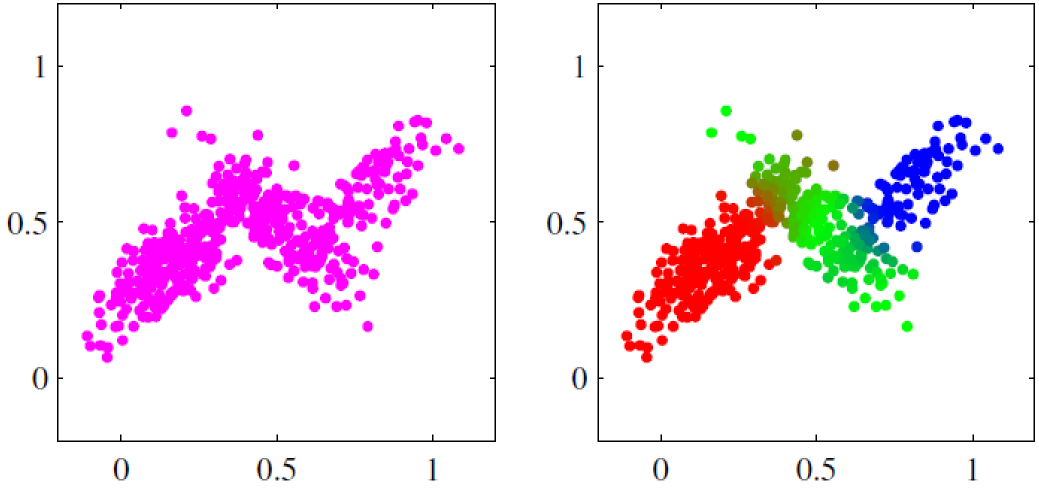


Mixture of Gaussians: Learning and Inference using Expectation-Maximization (EM)

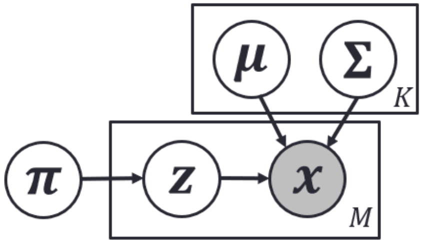


Learning: Given observed data $\boldsymbol{x_m}$, learn the parameters $\boldsymbol{\pi}, \boldsymbol{\mu_k}, \boldsymbol{\Sigma_k}$

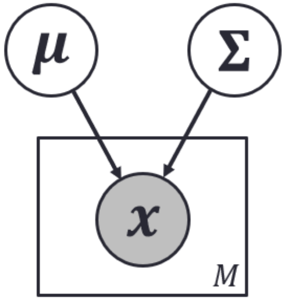
1. **Initialize** the means $\boldsymbol{\mu_k}$, covariances $\boldsymbol{\Sigma_k}$ and mixing coefficients $\boldsymbol{\pi}$.
2. **E Step.** Infer the expectation (distribution) of $\boldsymbol{z_m}$, denoted as $\gamma\left(\boldsymbol{z_m^{(k)}}\right)$, given the current parameters $\boldsymbol{\pi}$, $\boldsymbol{\mu_k}$ and $\boldsymbol{\Sigma_k}$.
3. **M Step.** Update the parameters $\boldsymbol{\pi}, \boldsymbol{\mu_k}, \boldsymbol{\Sigma_k}$ given the current $\gamma\left(\boldsymbol{z_m^{(k)}}\right)$.
4. Iterate between **E step** and **M step** until convergence.



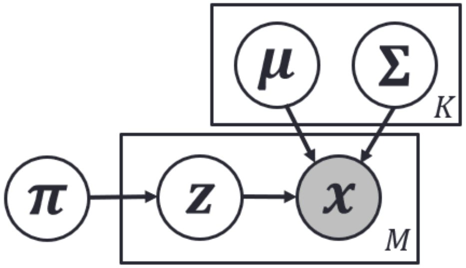
Mixture of Gaussians: Visualization



Summary on Probabilistic Graphical Models

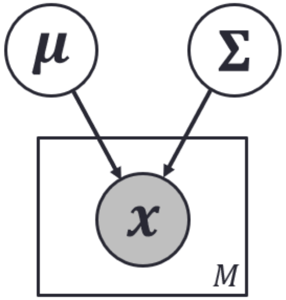


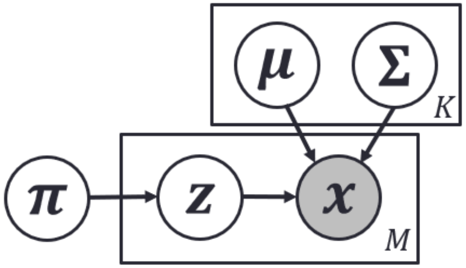
Gaussian Distribution



Mixture of Gaussians

Summary on Learning and Inference Algorithms

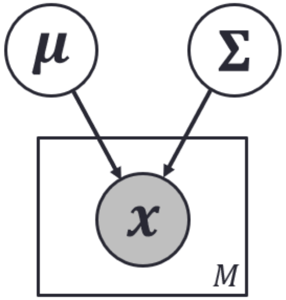




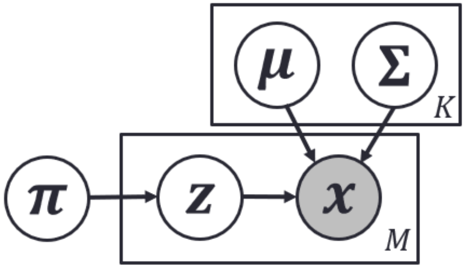
Maximum Likelihood Estimation (MLE)

EM

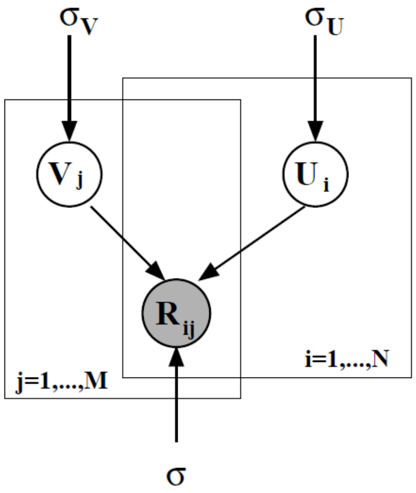
Summary on Probabilistic Graphical Models



Gaussian Distribution

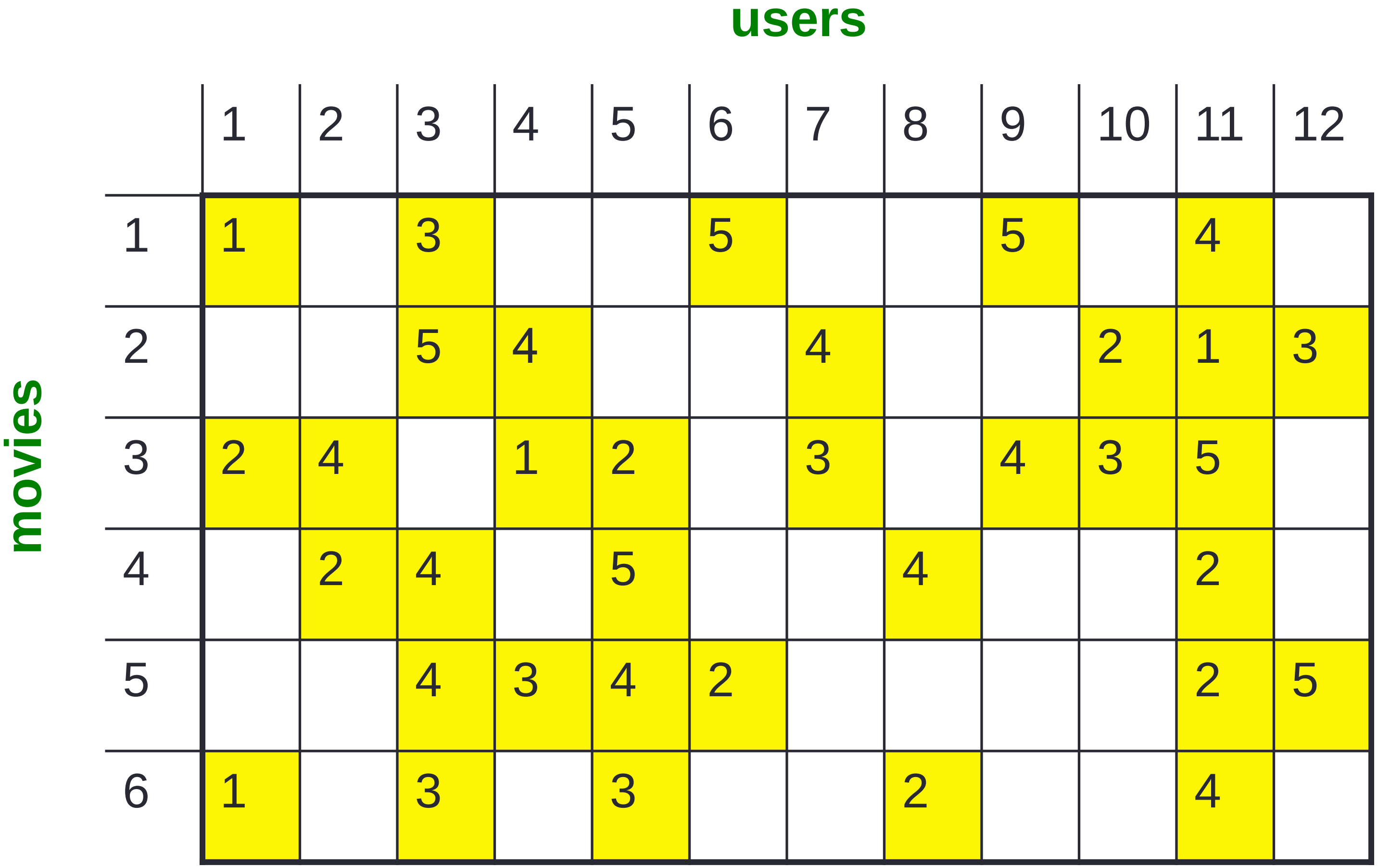


Mixture of Gaussians



Probabilistic Matrix Factorization (PMF)

The Rating Prediction Problem for Recommender Systems



users

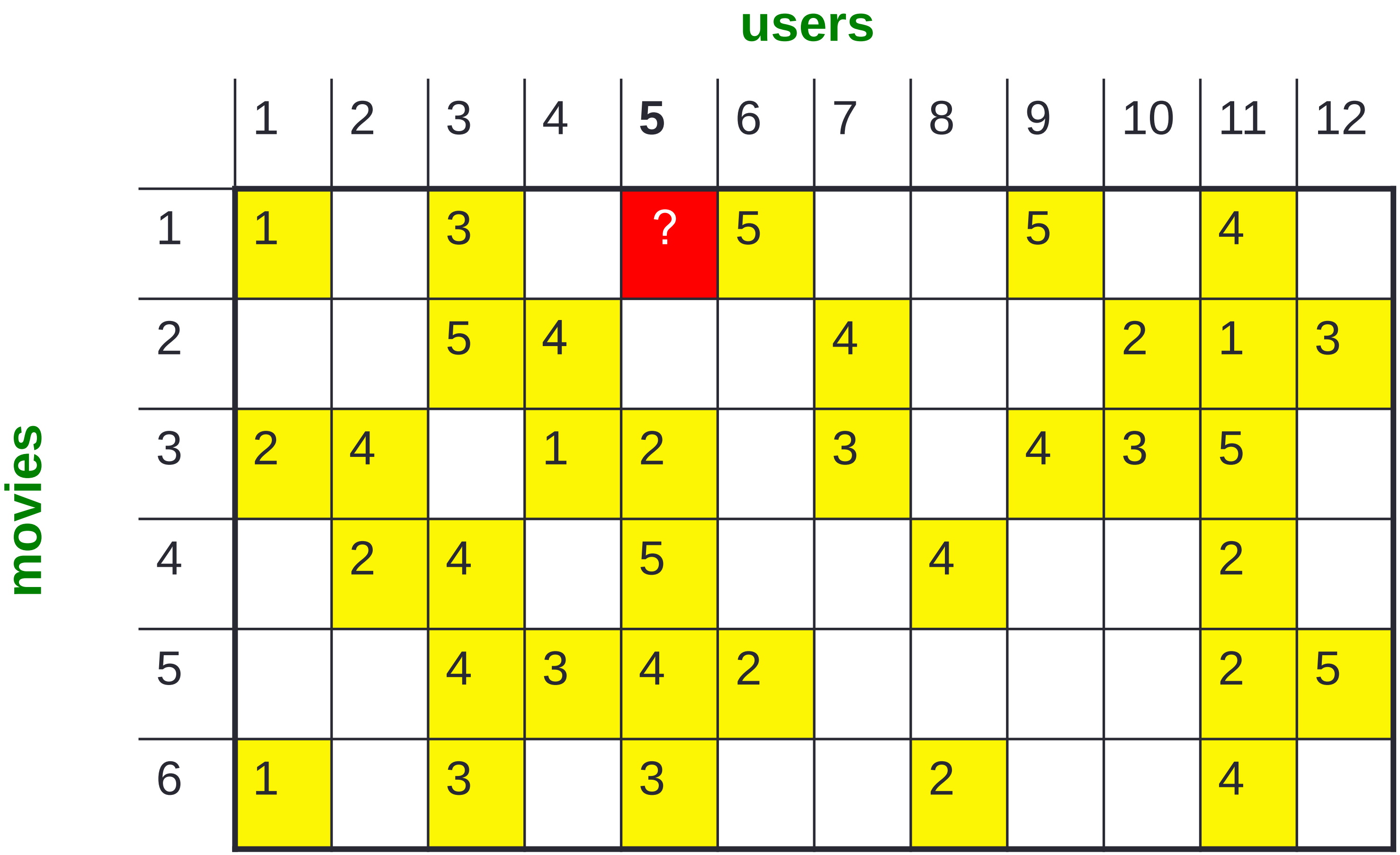
movies

	1	2	3	4	5	6	7	8	9	10	11	12
1	1		3			5			5		4	
2			5	4			4			2	1	3
3	2	4		1	2		3		4	3	5	
4		2	4		5			4			2	
5			4	3	4	2					2	5
6	1		3		3			2			4	

☐ - unknown rating

🟨 - rating between 1 to 5

The Rating Prediction Problem for Recommender Systems

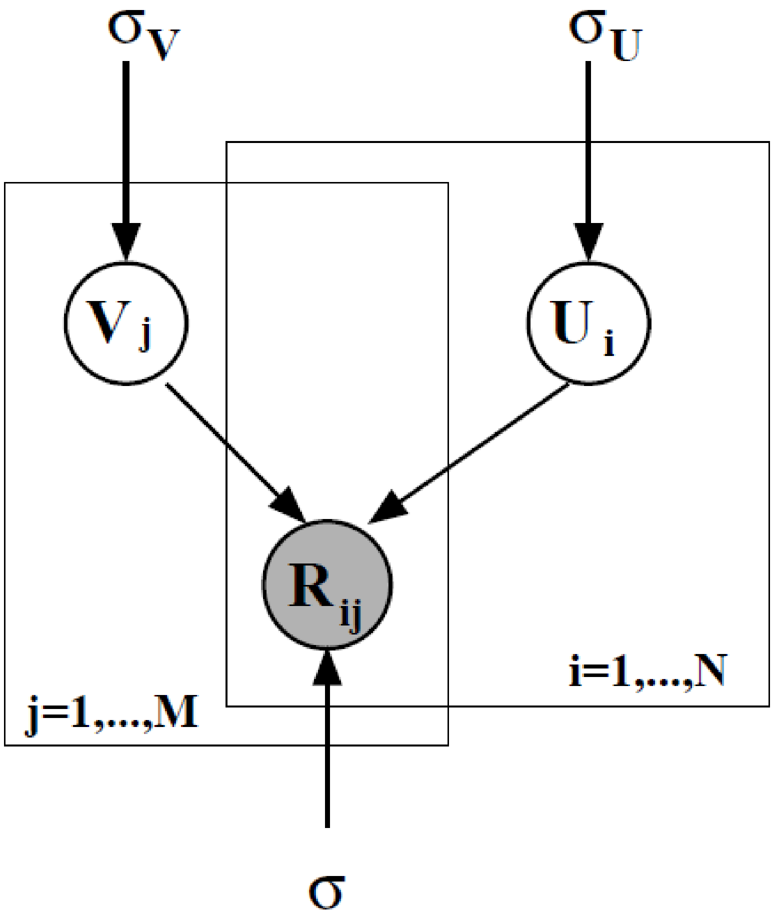


users

movies

	1	2	3	4	**5**	6	7	8	9	10	11	12
1	1		3		?	5			5		4	
2			5	4			4			2	1	3
3	2	4		1	2		3		4	3	5	
4		2	4		5			4			2	
5			4	3	4	2					2	5
6	1		3		3			2			4	

Probabilistic Matrix Factorization: Generative Process



$\sigma_U, \sigma_V, \sigma$ are hyperparameters

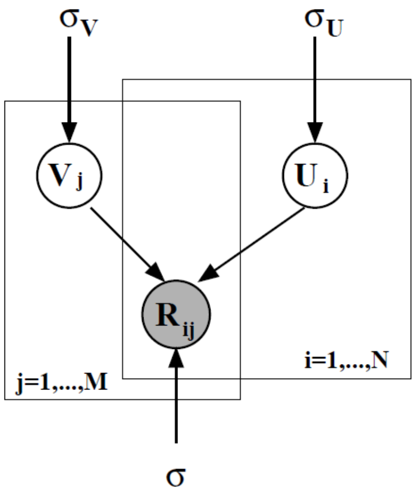
Notation

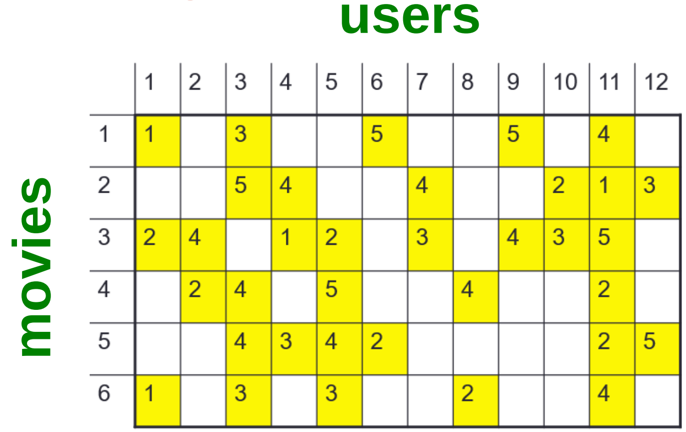
Latent user vector for user i: $U_i \in R^D$
Latent item vector for item j: $V_j \in R^D$
N users: $U \in R^{D\times N}$, M items: $V \in R^{D\times M}$
Rating that user i gives item j: $R_{ij} \in R$

Generative Process

1. For each user i:
    Generate user vector $U_i \sim N(U_i|0, \sigma_U I)$
2. For each item j:
    Generate item vector $V_j \sim N(V_j|0, \sigma_V I)$
3. For each user-item pair (i, j):
    Generate rating $R_{ij} \sim N(R_{ij}|U_i^T V_j, \sigma)$

Probabilistic Matrix Factorization: Generative Process - Factorization





	1	2	3	4	5	6	7	8	9	10	11	12
1	1		3			5			5		4	
2			5	4			4			2	1	3
3	2	4		1	2		3		4	3	5	
4		2	4		5			4			2	
5			4	3	4	2					2	5
6	1		3		3			2			4	

Generative Process

$$p(R,U,V|\sigma,\sigma_U,\sigma_V) = p(U|\sigma_U)p(V|\sigma_V)p(R|U,V,\sigma)$$

1. For each user i:
   Generate user vector $U_i \sim N(U_i|0,\sigma_U I)$

$$p(U|\sigma_U^2) = \prod_{i=1}^{N} \mathcal{N}(U_i|0,\sigma_U^2\mathbf{I})$$

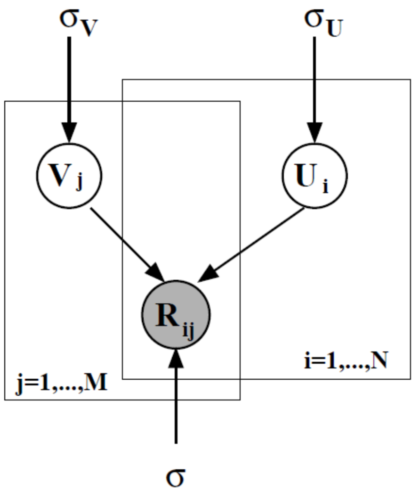
2. For each item j:
   Generate item vector $V_j \sim N(V_j|0,\sigma_V I)$

$$p(V|\sigma_V^2) = \prod_{j=1}^{M} \mathcal{N}(V_j|0,\sigma_V^2\mathbf{I})$$

3. For each user-item pair (i,j):
   Generate rating $R_{ij} \sim N(R_{ij}|U_i^T V_j,\sigma)$

$$p(R|U,V,\sigma^2) = \prod_{i=1}^{N}\prod_{j=1}^{M}\left[\mathcal{N}(R_{ij}|U_i^T V_j,\sigma^2)\right]^{I_{ij}}$$

Probabilistic Matrix Factorization: Learning and Inference



Notation

Latent user vector for user i: $U_i \in R^D$
Latent item vector for item j: $V_j \in R^D$
N users: $U \in R^{D \times N}$, M items: $V \in R^{D \times M}$
Rating that user i gives item j: $R_{ij} \in R$

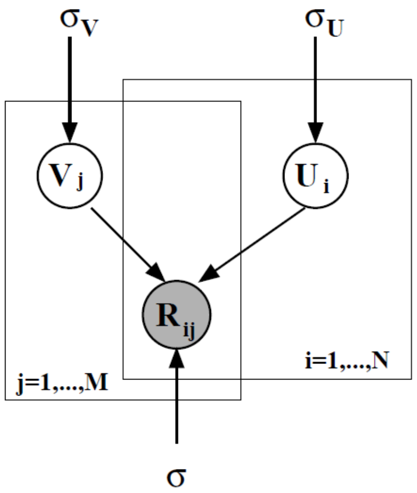
Learning: Given observed data, learn the unknown global parameters.

Not applicable since $\sigma_U, \sigma_V, \sigma$ are fixed (hyperparameters)

Inference: Given observed data and (hyper)parameters, infer the latent variables.

R_{ij} $\sigma_U, \sigma_V, \sigma$ U_i, V_j

Probabilistic Matrix Factorization: Maximum A Posteriori (MAP) Inference



Notation

Latent user vector for user i: $U_i \in R^D$
Latent item vector for item j: $V_j \in R^D$
N users: $U \in R^{D\times N}$, M items: $V \in R^{D\times M}$
Rating that user i gives item j: $R_{ij} \in R$

Posterior distribution of U and V:

$$p(U,V\big|R,\sigma^2,\sigma_V^2,\sigma_U^2) = \frac{p(U,V,R|\sigma^2,\sigma_V^2,\sigma_U^2)}{p(R|\sigma^2,\sigma_V^2,\sigma_U^2)} = \frac{p(U|\sigma_U)p(V|\sigma_V)p(R|U,V,\sigma)}{p(R|\sigma^2,\sigma_V^2,\sigma_U^2)}$$

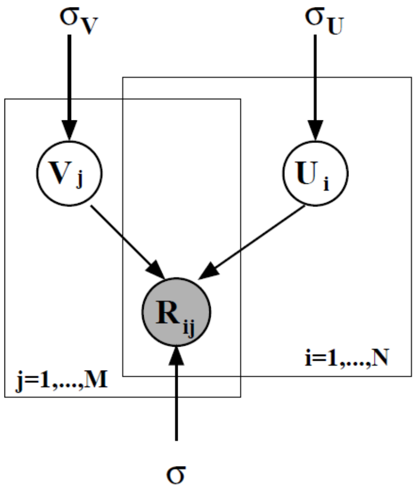
Constant C

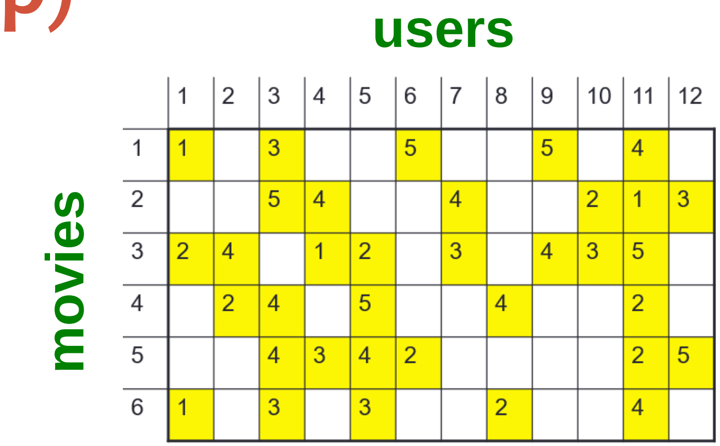
The **log** of the posterior distribution of U and V becomes:

$$\log p(U,V\big|R,\sigma^2,\sigma_V^2,\sigma_U^2) = \log p(R|U,V,\sigma) + \log p(U|\sigma_U) + \log p(V|\sigma_V) + C$$

Probabilistic Matrix Factorization: Generative Process (Recap)

$$f_{\mathbf{X}}(x_1,\ldots,x_k) = \frac{\exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)}{\sqrt{(2\pi)^k|\boldsymbol{\Sigma}|}}$$





	1	2	3	4	5	6	7	8	9	10	11	12
1	1		3			5			5		4	
2			5	4			4			2	1	3
3	2	4		1	2		3		4	3	5	
4		2	4		5			4			2	
5			4	3	4	2					2	5
6	1		3		3			2			4	

Generative Process

$$p(R,U,V|\sigma,\sigma_U,\sigma_V) \\ = p(U|\sigma_U)p(V|\sigma_V)p(R|U,V,\sigma)$$

1. For each user i:
   Generate user vector $U_i \sim N(U_i|0,\sigma_U I)$

$$p(U|\sigma_U^2) = \prod_{i=1}^{N} \mathcal{N}(U_i|0,\sigma_U^2\mathbf{I})$$

2. For each item j:
   Generate item vector $V_j \sim N(V_j|0,\sigma_V I)$

$$p(V|\sigma_V^2) = \prod_{j=1}^{M} \mathcal{N}(V_j|0,\sigma_V^2\mathbf{I})$$

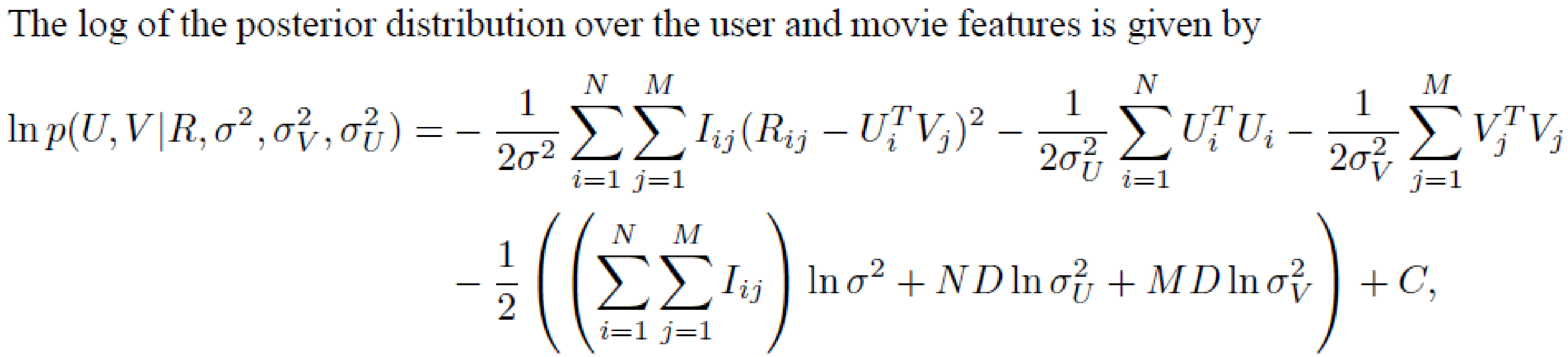
3. For each user-item pair (i,j):
   Generate rating $R_{ij} \sim N(R_{ij}|U_i^T V_j,\sigma)$

$$p(R|U,V,\sigma^2) = \prod_{i=1}^{N}\prod_{j=1}^{M}\left[\mathcal{N}(R_{ij}|U_i^T V_j,\sigma^2)\right]^{I_{ij}}$$

Probabilistic Matrix Factorization: Maximum A Posteriori (MAP) Inference

The **log** of the posterior distribution of U and V becomes:

$$\log p(U,V|R,\sigma^2,\sigma_V^2,\sigma_U^2) = \log p(R|U,V,\sigma) + \log p(U|\sigma_U) + \log p(V|\sigma_V) + C$$

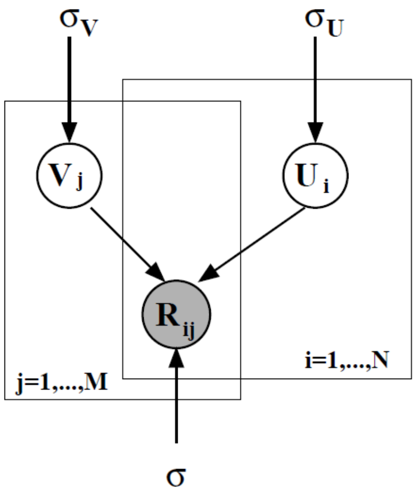


The log of the posterior distribution over the user and movie features is given by

$$\ln p(U,V|R,\sigma^2,\sigma_V^2,\sigma_U^2) = -\frac{1}{2\sigma^2}\sum_{i=1}^{N}\sum_{j=1}^{M} I_{ij}(R_{ij}-U_i^T V_j)^2 - \frac{1}{2\sigma_U^2}\sum_{i=1}^{N} U_i^T U_i - \frac{1}{2\sigma_V^2}\sum_{j=1}^{M} V_j^T V_j$$
$$-\frac{1}{2}\left(\left(\sum_{i=1}^{N}\sum_{j=1}^{M} I_{ij}\right)\ln\sigma^2 + ND\ln\sigma_U^2 + MD\ln\sigma_V^2\right) + C,$$

$I_{ij}, \sigma, \sigma_U, \sigma_V, C$ are constants.

Probabilistic Matrix Factorization: Maximum A Posteriori (MAP) Inference



Maximizing the log-posterior over item vectors V_j and user vectors U_i when fixing the hyperparameters (i.e. the observation noise variance σ and prior variances σ_U, σ_V) is equivalent to minimizing the **sum-of-squared-errors** objective function with **quadratic regularization terms**:

$$E = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{M} I_{ij}\left(R_{ij} - U_i^T V_j\right)^2 + \frac{\lambda_U}{2}\sum_{i=1}^{N} \parallel U_i \parallel_{Fro}^2 + \frac{\lambda_V}{2}\sum_{j=1}^{M} \parallel V_j \parallel_{Fro}^2,$$

where $\lambda_U = \sigma^2/\sigma_U$, $\lambda_V = \sigma^2/\sigma_V$, and $||\cdot||_{Fro}$ denotes the Frobenius norm.

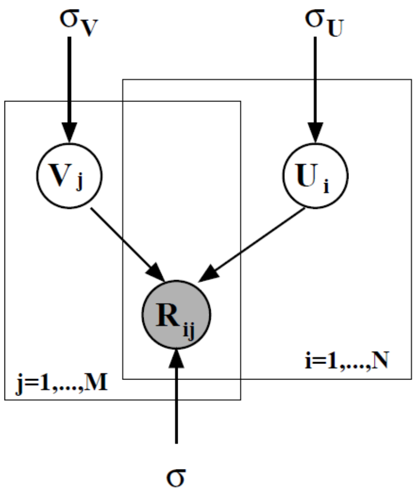
Probabilistic Matrix Factorization: Maximum A Posteriori (MAP) Inference

$$E = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{M} I_{ij}\left(R_{ij} - U_i^T V_j\right)^2 + \frac{\lambda_U}{2}\sum_{i=1}^{N} \| U_i \|_{Fro}^2 + \frac{\lambda_V}{2}\sum_{j=1}^{M} \| V_j \|_{Fro}^2,$$

How to find the U_i and V_j that minimize E? Use gradient descent!

```
Initialize $U_i$ and $V_j$
For each iteration t = 1:T do
    For each user i = 1:N do
        $U_i = U_i - \rho_t \frac{\partial E}{\partial U_i}$
    For each item j = 1:M do
        $V_j = V_j - \rho_t \frac{\partial E}{\partial V_j}$
```

Probabilistic Matrix Factorization: Learning or Inference?



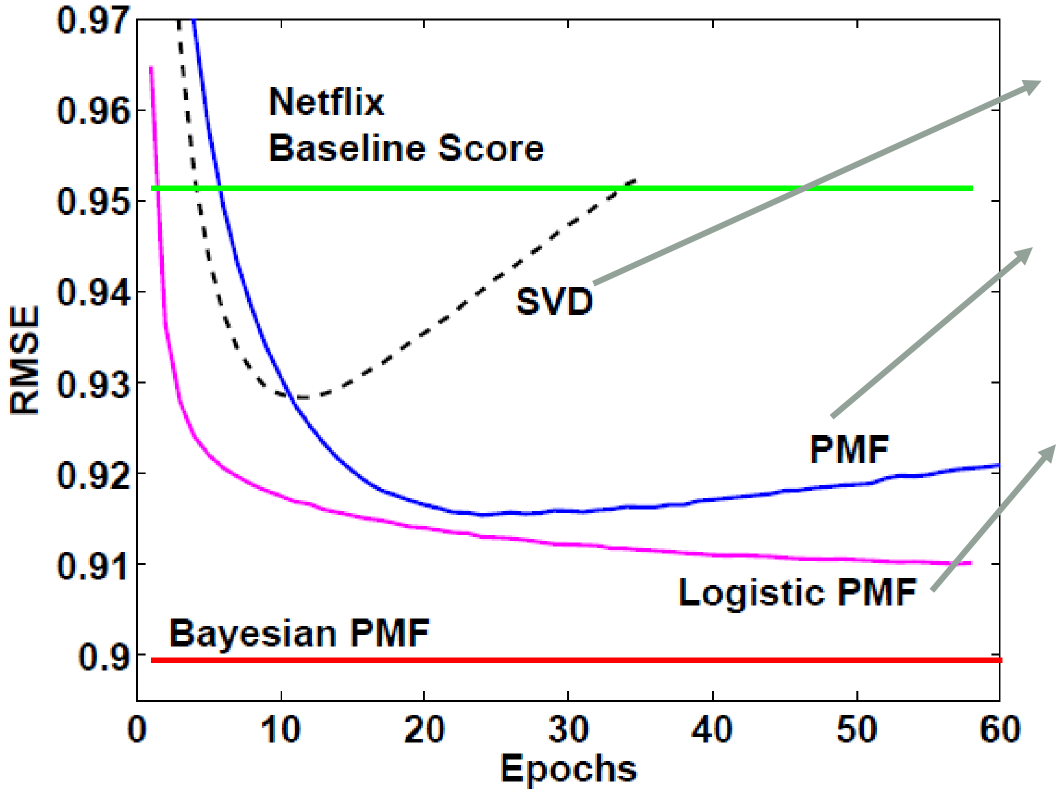
(Global) parameters σ_V, σ_U, and σ are fixed (we treat them as hyperparameters that are manually set).

We are trying to estimate (local) **latent variables** V_j and U_i.

Answer: Inference.

Probabilistic Matrix Factorization (PMF): Experimental Results

Dataset: Netflix.
Size: 100M ratings, 480K users, 17K movies.



SVD: Very easy to overfit

PMF: Better performance and harder to overfit

Logistic PMF: Even better performance

(RMSE: Difference between predicted and ground-truth ratings.)

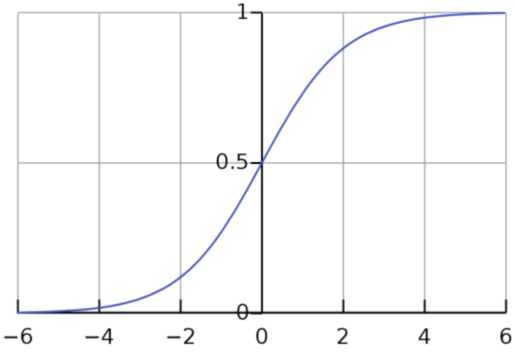
Logistic PMF:
Maximum A Posteriori (MAP) Inference

$$E = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{M} I_{ij}(R_{ij} - \underline{U_i^T V_j})^2 + \frac{\lambda_U}{2}\sum_{i=1}^{N} \| U_i \|_{Fro}^2 + \frac{\lambda_V}{2}\sum_{j=1}^{M} \| V_j \|_{Fro}^2,$$

Use a logistic function on the inner product

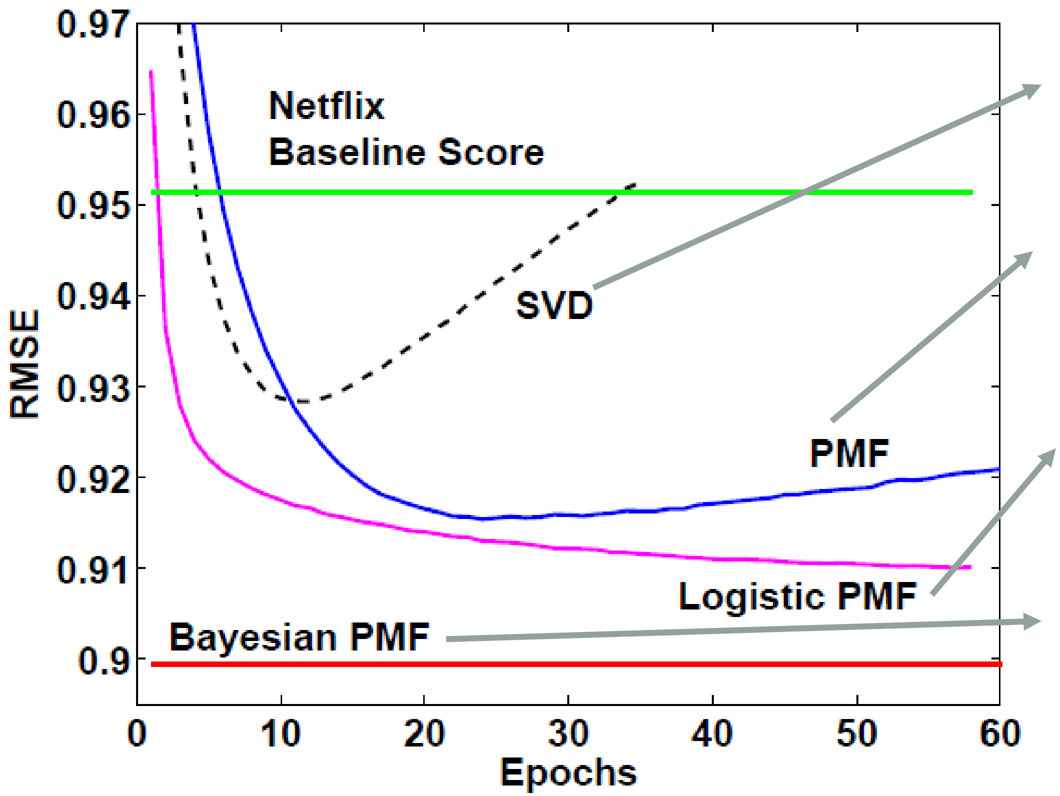
$$U_i^T V_j \rightarrow g(U_i^T V_j),$$

where the logistic (sigmoid) function $g(x) = 1/(1 + \exp(-x))$



Probabilistic Matrix Factorization (PMF): Experimental Results

Dataset: Netflix.
Size: 100M ratings, 480K users, 17K movies.



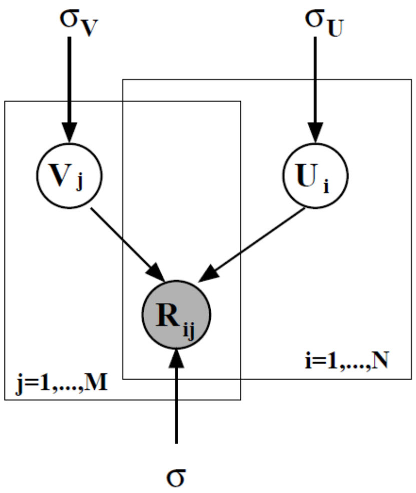
SVD: Very easy to overfit

PMF: Better performance and harder to overfit

Logistic PMF: Even better performance

Bayes PMF: Best performance

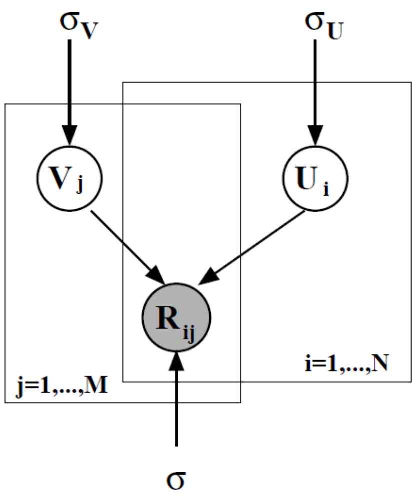
Probabilistic Matrix Factorization (PMF)



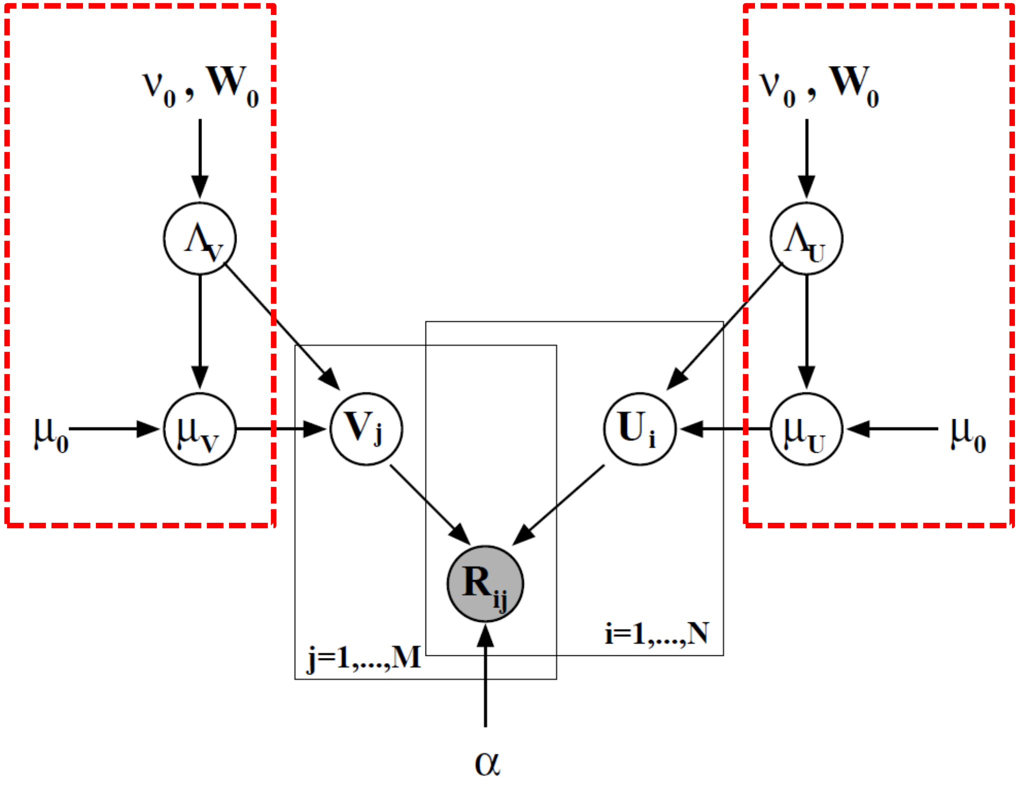
(Global) parameters σ_V, σ_U, and σ are **fixed** (we treat them as hyperparameters that are manually set).

Can we make the parameters learnable?

Bayesian Probabilistic Matrix Factorization (BPMF)



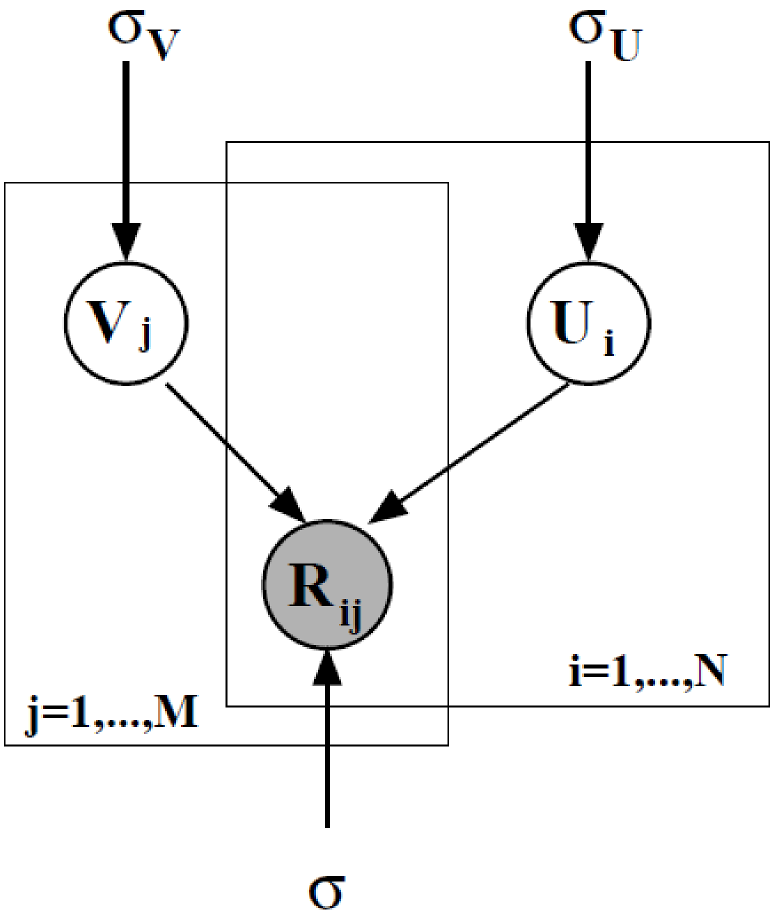
PMF



Bayesian PMF

$N(\mu_V, \Lambda_V^{-1})$: Λ_V is the precision matrix, Λ_V^{-1} is the covariance matrix

Probabilistic Matrix Factorization: Generative Process (Recap)

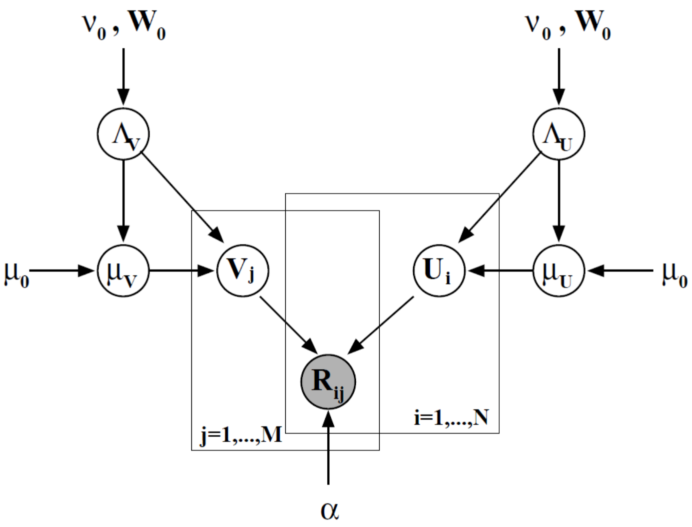


Generative Process

1. For each user i:
   Generate user vector $U_i \sim N(U_i|0, \sigma_U I)$
2. For each item j:
   Generate item vector $V_j \sim N(V_j|0, \sigma_V I)$
3. For each user-item pair (i, j):
   Generate rating $R_{ij} \sim N(R_{ij}|U_i^T V_j, \sigma I)$

Bayesian Probabilistic Matrix Factorization (BPMF): Generative Process

Bayesian PMF



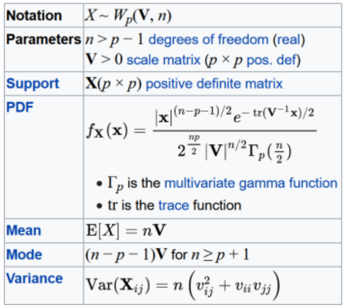
Generative Process

1. Generate user precision matrix $\Lambda_U \sim W(\Lambda_U | W_0, \nu_0)$
2. Generate user mean $\mu_U \sim N(\mu_U | \mu_0, (\beta_0 \Lambda_U)^{-1})$
3. For each user i:
   Generate user vector $U_i \sim N(U_i | \mu_U, \Lambda_U^{-1})$

4. Generate item precision matrix $\Lambda_V \sim W(\Lambda_V | W_0, \nu_0)$
5. Generate item mean $\mu_V \sim N(\mu_V | \mu_0, (\beta_0 \Lambda_V)^{-1})$
6. For each item j:
   Generate item vector $V_j \sim N(V_j | \mu_V, \Lambda_V^{-1})$

7. For each user-item pair (i, j):
   Generate rating $R_{ij} \sim N(R_{ij} | U_i^T V_j, \alpha^{-1})$

Wishart distribution

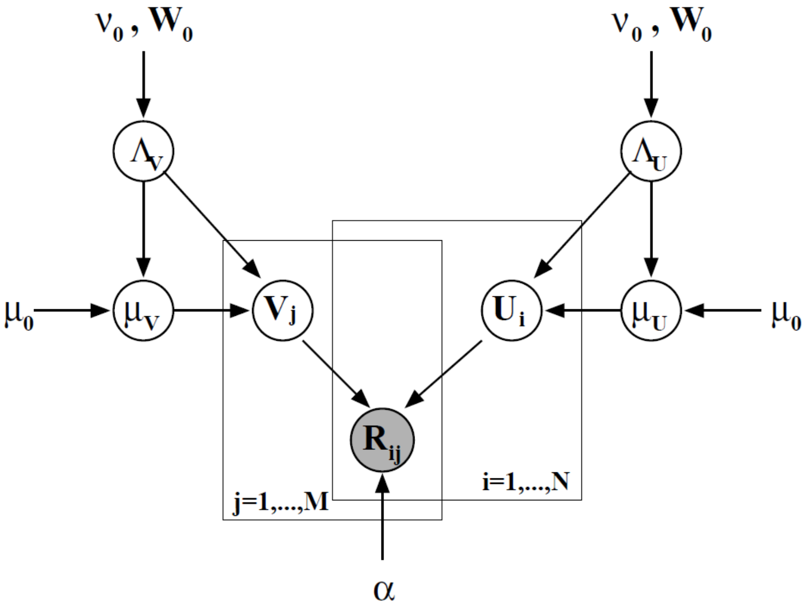


Notation	$X \sim W_p(\mathbf{V}, n)$
Parameters	$n > p - 1$ degrees of freedom (real) $\mathbf{V} > 0$ scale matrix ($p \times p$ pos. def)
Support	$\mathbf{X}(p \times p)$ positive definite matrix
PDF	$f_{\mathbf{X}}(\mathbf{x}) = \frac{\lvert\mathbf{x}\rvert^{(n-p-1)/2} e^{-\operatorname{tr}(\mathbf{V}^{-1}\mathbf{x})/2}}{2^{\frac{np}{2}} \lvert\mathbf{V}\rvert^{n/2} \Gamma_p(\frac{n}{2})}$ • Γ_p is the multivariate gamma function • tr is the trace function
Mean	$\operatorname{E}[X] = n\mathbf{V}$
Mode	$(n - p - 1)\mathbf{V}$ for $n \geq p + 1$
Variance	$\operatorname{Var}(\mathbf{X}_{ij}) = n\left(v_{ij}^2 + v_{ii}v_{jj}\right)$

In a Gaussian distribution, $N(\mu, \Sigma)$,
Σ is the covariance matrix
and Σ^{-1} is the precision matrix

Bayesian Probabilistic Matrix Factorization (BPMF): Learning and Inference

Bayesian PMF



(Global) parameters: $\Theta_U = \{\Lambda_U, \mu_U\}$, $\Theta_V = \{\Lambda_V, \mu_V\}$

(Local) latent variables: U_i, V_j

Hyperparameters: $\Theta_0 = \{\nu_0, W_0, \mu_0\}$

Learning: Given the data R_{ij}, estimate the optimal parameters $\Lambda_U, \mu_U, \Lambda_V, \mu_V$

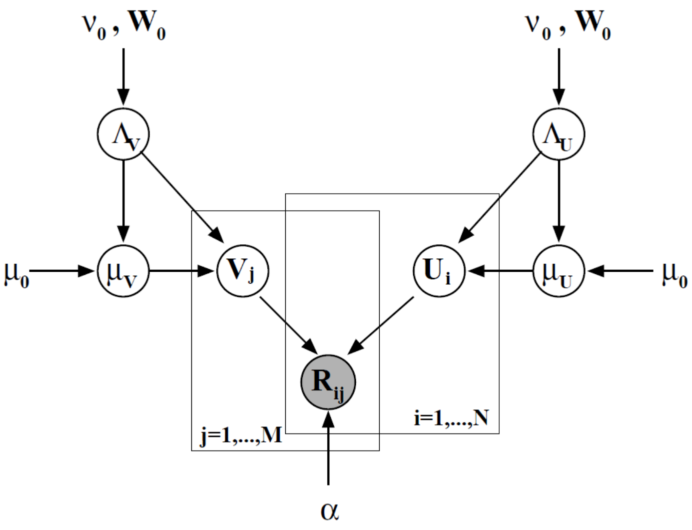
Inference: Given the data R_{ij} and the

How to perform learning and inference?

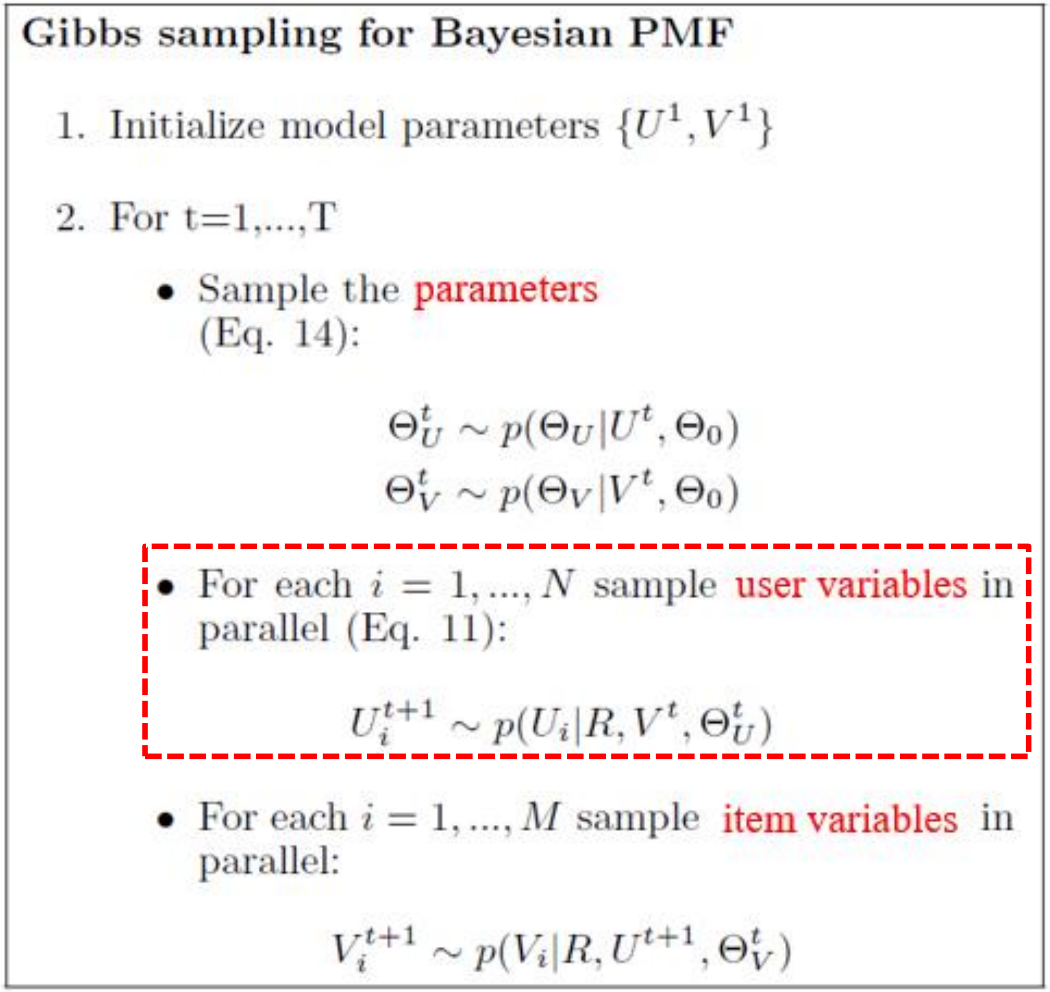
and item U_i, V_j

Bayesian Probabilistic Matrix Factorization (BPMF): Learning and Inference Using Gibbs Sampling

Bayesian PMF



Hyperparameters: $\Theta_0 = \{\nu_0, W_0, \mu_0\}$
(Global) parameters: $\Theta_U = \{\Lambda_U, \mu_U\}$, $\Theta_V = \{\Lambda_V, \mu_V\}$
(Local) latent variables: U_i, V_j



Gibbs sampling for Bayesian PMF

1. Initialize model parameters $\{U^1, V^1\}$
2. For t=1,...,T
    - Sample the parameters (Eq. 14):

$$\Theta_U^t \sim p(\Theta_U | U^t, \Theta_0)$$
$$\Theta_V^t \sim p(\Theta_V | V^t, \Theta_0)$$

    - For each $i = 1, ..., N$ sample user variables in parallel (Eq. 11):

$$U_i^{t+1} \sim p(U_i | R, V^t, \Theta_U^t)$$

    - For each $i = 1, ..., M$ sample item variables in parallel:

$$V_i^{t+1} \sim p(V_i | R, U^{t+1}, \Theta_V^t)$$

Bayesian Probabilistic Matrix Factorization (BPMF): Learning and Inference Using Gibbs Sampling

- For each $i = 1, ..., N$ sample user variables in parallel (Eq. 11):

$$U_i^{t+1} \sim p(U_i|R, V^t, \Theta_U^t)$$

Updating user i' variable U_i

$$p(U_i|R, V, \Theta_U, \alpha) = \boxed{\mathcal{N}\left(U_i|\mu_i^*, [\Lambda_i^*]^{-1}\right)}$$
$$\sim \prod_{j=1}^{M} \left[\mathcal{N}(R_{ij}|U_i^T V_j, \alpha^{-1})\right]^{I_{ij}} p(U_i|\mu_U, \Lambda_U),$$

where

$$\Lambda_i^* = \Lambda_U + \alpha \boxed{\sum_{j=1}^{M} \left[V_j V_j^T\right]^{I_{ij}}}$$
$$\mu_i^* = [\Lambda_i^*]^{-1} \left(\alpha \sum_{j=1}^{M} \left[V_j R_{ij}\right]^{I_{ij}} + \Lambda_U \mu_U \right)$$

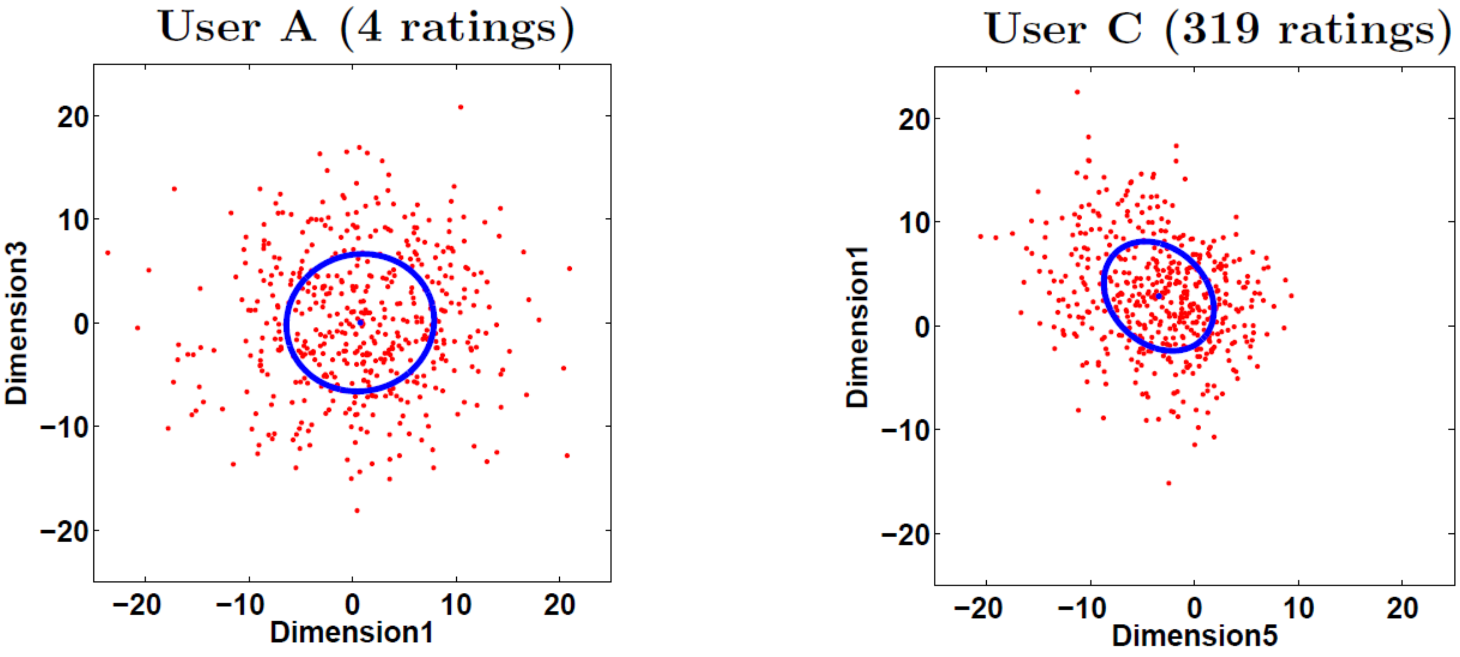
$I_{ij} = 1$ if user i rated movie j
$I_{ij} = 0$ if user i did not rate movie j

User i rated more movies
→ More $I_{ij} = 1$
→ This term gets larger
→ The precision matrix Λ_i^* gets larger
→ The covariance matrix $[\Lambda_i^*]^{-1}$gets smaller
→ The model is more confident about the distribution.

Sampling User i's Latent Variable U_i

$$\Lambda_i^* = \Lambda_U + \alpha \boxed{\sum_{j=1}^{M} \left[V_j V_j^T\right]^{I_{ij}}}$$

$$\mu_i^* = [\Lambda_i^*]^{-1}\left(\alpha \sum_{j=1}^{M} \left[V_j R_{ij}\right]^{I_{ij}} + \Lambda_U \mu_U\right)$$



The two dimensions with the highest variance are shown for two users

Bayesian Probabilistic Matrix Factorization (BPMF): Learning and Inference Using Gibbs Sampling

- For each $i = 1, ..., N$ sample user variables in parallel (Eq. 11):

$$U_i^{t+1} \sim p(U_i | R, V^t, \Theta_U^t)$$

Updating user i' variable U_i

$$p(U_i | R, V, \Theta_U, \alpha) = \boxed{\mathcal{N}\left(U_i | \mu_i^*, [\Lambda_i^*]^{-1}\right)}$$

$$\sim \prod_{j=1}^{M} \left[\mathcal{N}(R_{ij} | U_i^T V_j, \alpha^{-1}) \right]^{I_{ij}} p(U_i | \mu_U, \Lambda_U),$$

where

$$\Lambda_i^* = \Lambda_U + \alpha \sum_{j=1}^{M} \left[V_j V_j^T \right]^{I_{ij}}$$

$$\mu_i^* = [\Lambda_i^*]^{-1} \left(\alpha \boxed{\sum_{j=1}^{M} \left[V_j R_{ij} \right]^{I_{ij}}} + \Lambda_U \mu_U \right)$$

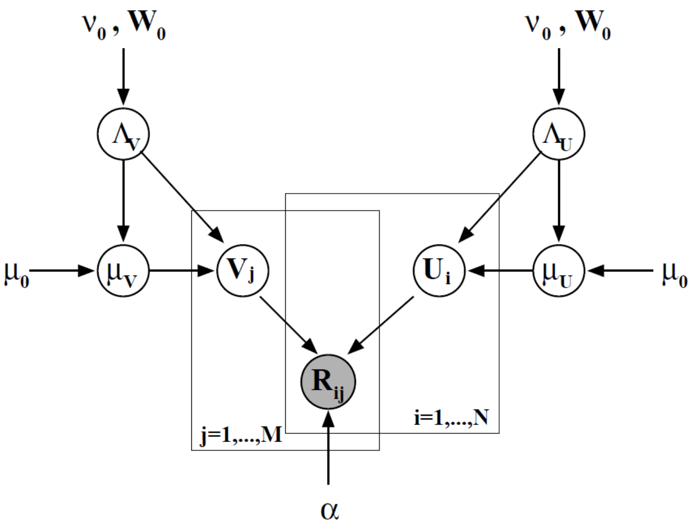
Weighted average of all the item latent variables V_j

The weight for item j's variable V_j is the rating user i gives item j, R_{ij}

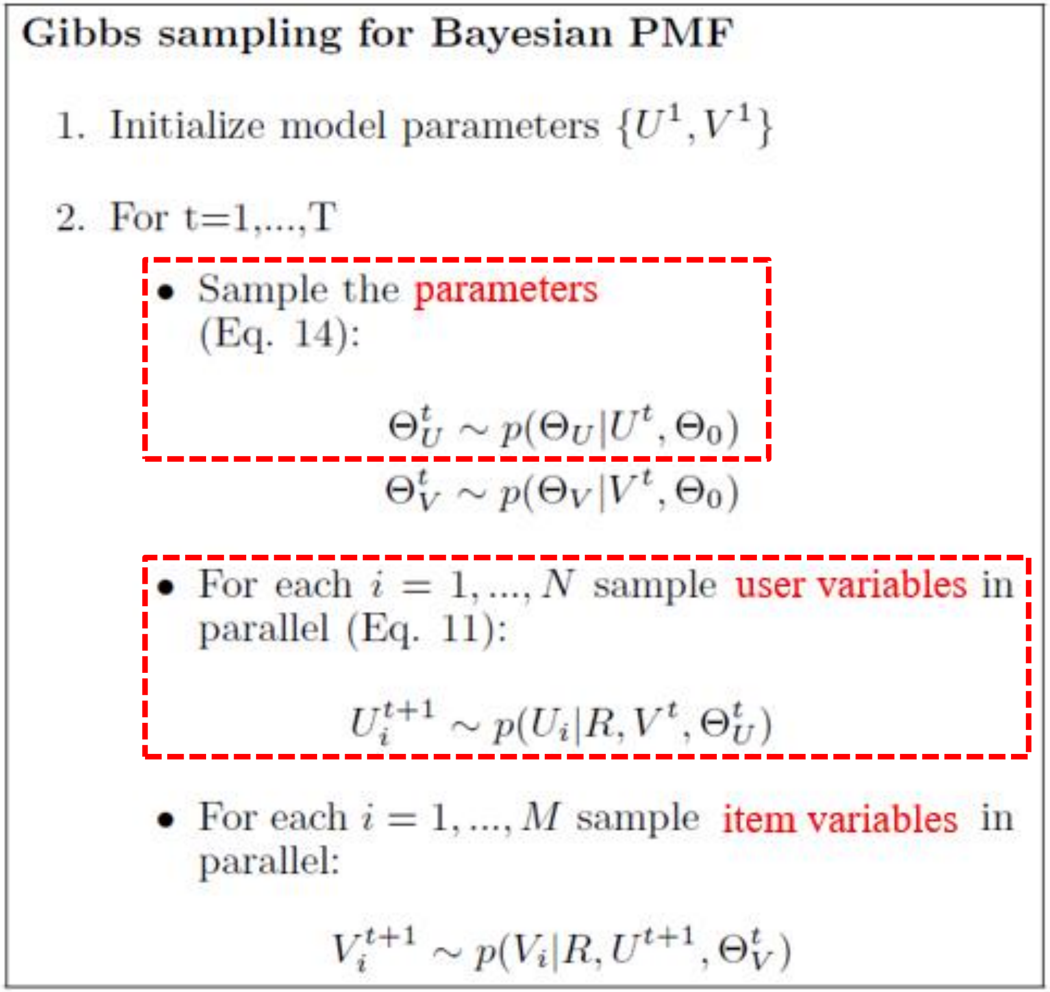
An item j is ignored if user i did not rate it ($I_{ij} = 0$)

Bayesian Probabilistic Matrix Factorization (BPMF): Learning and Inference Using Gibbs Sampling

Bayesian PMF



Hyperparameters: $\Theta_0 = \{\nu_0, W_0, \mu_0\}$
(Global) parameters: $\Theta_U = \{\Lambda_U, \mu_U\}$, $\Theta_V = \{\Lambda_V, \mu_V\}$
(Local) latent variables: U_i, V_j



Gibbs sampling for Bayesian PMF

1. Initialize model parameters $\{U^1, V^1\}$
2. For t=1,...,T
   - Sample the parameters (Eq. 14):
     $$\Theta_U^t \sim p(\Theta_U | U^t, \Theta_0)$$
     $$\Theta_V^t \sim p(\Theta_V | V^t, \Theta_0)$$
   - For each $i = 1, ..., N$ sample user variables in parallel (Eq. 11):
     $$U_i^{t+1} \sim p(U_i | R, V^t, \Theta_U^t)$$
   - For each $i = 1, ..., M$ sample item variables in parallel:
     $$V_i^{t+1} \sim p(V_i | R, U^{t+1}, \Theta_V^t)$$

Bayesian Probabilistic Matrix Factorization (BPMF): Learning and Inference Using Gibbs Sampling

- Sample the parameters (Eq. 14):

$$\Theta_U^t \sim p(\Theta_U | U^t, \Theta_0)$$

Updating (global) parameters $\boldsymbol{\Theta}_{\mathbf{U}} = \{\boldsymbol{\mu}_U, \boldsymbol{\Lambda}_{\mathbf{U}}\}$

$$p(\mu_U, \Lambda_U | U, \Theta_0) = \boxed{\mathcal{N}(\mu_U | \mu_0^*, (\beta_0^* \Lambda_U)^{-1})} \mathcal{W}(\Lambda_U | W_0^*, \nu_0^*),$$

where

$$\boxed{\mu_0^* = \frac{\beta_0 \mu_0 + N\bar{U}}{\beta_0 + N},} \quad \beta_0^* = \beta_0 + N, \quad \nu_0^* = \nu_0 + N,$$

$$[W_0^*]^{-1} = W_0^{-1} + N\bar{S} + \frac{\beta_0 N}{\beta_0 + N}(\mu_0 - \bar{U})(\mu_0 - \bar{U})^T$$

$$\bar{U} = \frac{1}{N}\sum_{i=1}^{N} U_i \qquad \bar{S} = \frac{1}{N}\sum_{i=1}^{N} U_i U_i^T.$$

Weighted average of μ_0 and $\bar{U}$ (μ_0 is a hyperparameter)

$\bar{U}$ is the average of all the user latent variables U_i

The weight for μ_0 is β_0 (μ_0 is a hyperparameter)

The weight for $\bar{U}$ is N (N is the number of users)

Bayesian Probabilistic Matrix Factorization (BPMF): Learning and Inference Using Gibbs Sampling

- Sample the parameters (Eq. 14):

$$\Theta_U^t \sim p(\Theta_U | U^t, \Theta_0)$$

Updating (global) parameters $\boldsymbol{\Theta}_{\mathbf{U}} = \{\boldsymbol{\mu}_U, \boldsymbol{\Lambda}_{\mathbf{U}}\}$

$$p(\mu_U, \Lambda_U | U, \Theta_0) = \boxed{\mathcal{N}(\mu_U | \mu_0^*, (\beta_0^* \Lambda_U)^{-1})} \mathcal{W}(\Lambda_U | W_0^*, \nu_0^*),$$

where

$$\mu_0^* = \frac{\beta_0 \mu_0 + N\bar{U}}{\beta_0 + N}, \quad \boxed{\beta_0^* = \beta_0 + N,} \quad \nu_0^* = \nu_0 + N,$$

$$[W_0^*]^{-1} = W_0^{-1} + N\bar{S} + \frac{\beta_0 N}{\beta_0 + N}(\mu_0 - \bar{U})(\mu_0 - \bar{U})^T$$

$$\bar{U} = \frac{1}{N}\sum_{i=1}^{N} U_i \qquad \bar{S} = \frac{1}{N}\sum_{i=1}^{N} U_i U_i^T.$$

β_0 is a hyperparameter (which is fixed)

N is the number of users

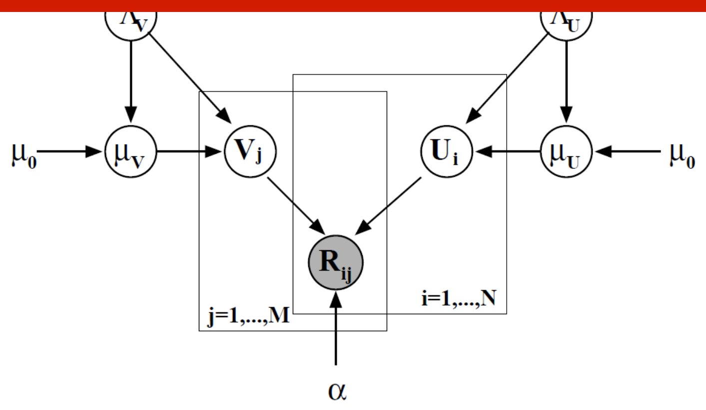
If we have more users, β_0^* will get larger,

the covariance $(\beta_0^* \Lambda_U)^{-1}$ will get smaller

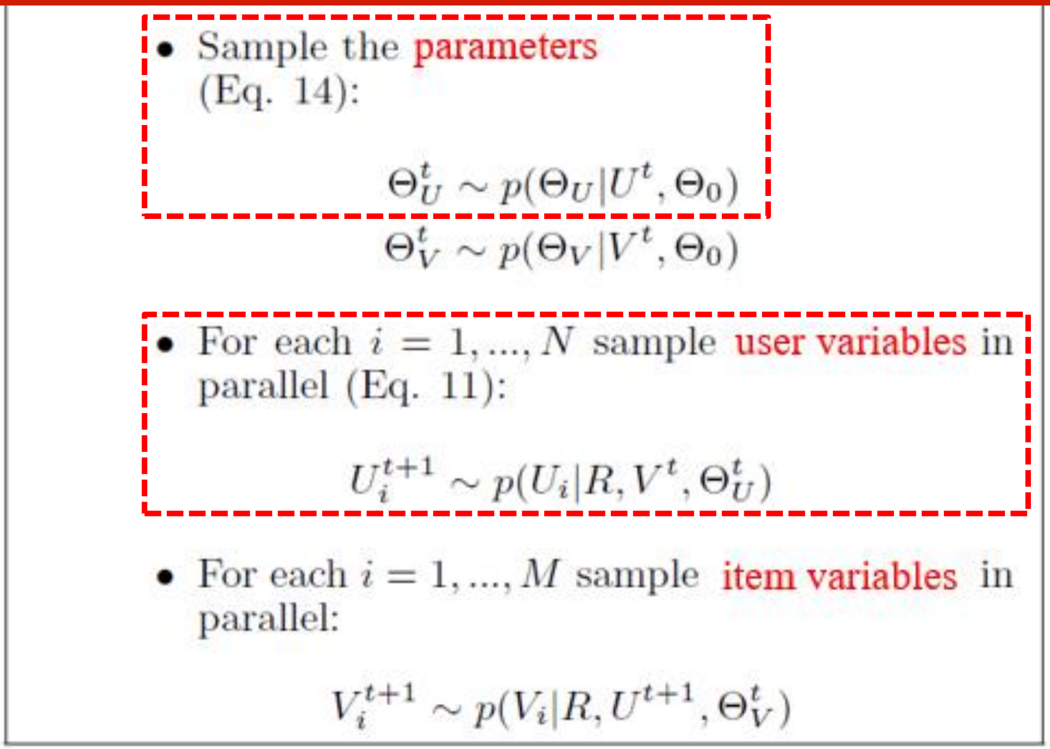
The model is more confident about the distribution on μ_U.

Bayesian Probabilistic Matrix Factorization (BPMF): Learning and Inference Using Gibbs Sampling

We can update the latent variables and parameters similarly on the item side.



Hyperparameters: $\Theta_0 = \{\nu_0, W_0, \mu_0\}$
(Global) parameters: $\Theta_U = \{\Lambda_U, \mu_U\}$, $\Theta_V = \{\Lambda_V, \mu_V\}$
(Local) latent variables: U_i, V_j



- Sample the parameters (Eq. 14):

$$\Theta_U^t \sim p(\Theta_U|U^t, \Theta_0)$$
$$\Theta_V^t \sim p(\Theta_V|V^t, \Theta_0)$$

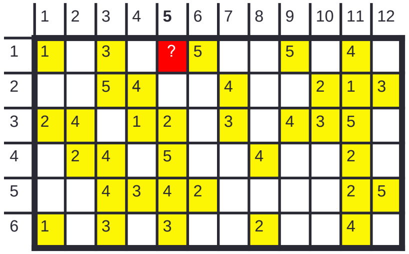
- For each $i = 1, ..., N$ sample user variables in parallel (Eq. 11):

$$U_i^{t+1} \sim p(U_i|R, V^t, \Theta_U^t)$$

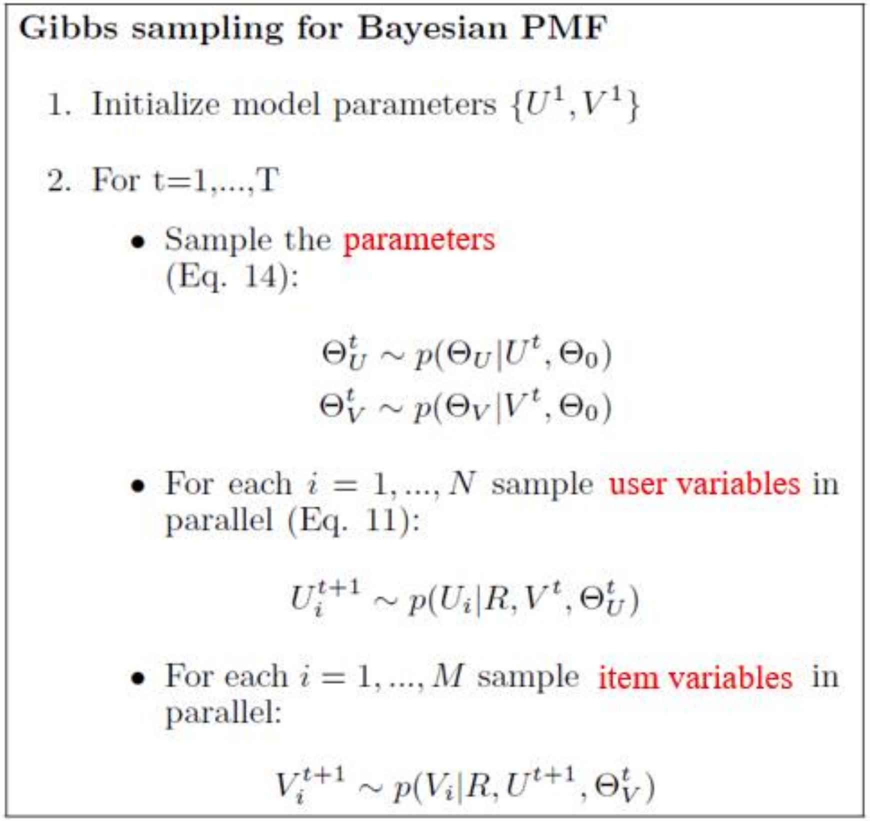
- For each $i = 1, ..., M$ sample item variables in parallel:

$$V_i^{t+1} \sim p(V_i|R, U^{t+1}, \Theta_V^t)$$

After Learning, How to Make Predictions



	1	2	3	4	5	6	7	8	9	10	11	12
1	1		3		?	5			5		4	
2			5	4			4			2	1	3
3	2	4		1	2		3		4	3	5	
4		2	4		5			4			2	
5			4	3	4	2					2	5
6	1		3		3			2			4	



Gibbs sampling for Bayesian PMF

```
1. Initialize model parameters $\{U^1, V^1\}$
2. For t=1,...,T
    • Sample the parameters (Eq. 14):
        $\Theta_U^t \sim p(\Theta_U | U^t, \Theta_0)$
        $\Theta_V^t \sim p(\Theta_V | V^t, \Theta_0)$
    • For each $i = 1, ..., N$ sample user variables in parallel (Eq. 11):
        $U_i^{t+1} \sim p(U_i | R, V^t, \Theta_U^t)$
    • For each $i = 1, ..., M$ sample item variables in parallel:
        $V_i^{t+1} \sim p(V_i | R, U^{t+1}, \Theta_V^t)$
```

$$p(R_{ij}^* | R, \Theta_0) \approx \frac{1}{K} \sum_{k=1}^{K} p(R_{ij}^* | U_i^{(k)}, V_j^{(k)})$$

These K samples $U_i^{(k)}, V_j^{(k)}$ are generated by running K additional iterations after the Gibbs sampling algorithm converges

PMF versus Bayesian PMF

PMF

Use MAP inference to get point estimate of U_i and V_j given the data R_{ij}

Variances σ_U, σ_V are fixed as hyperparameters

Easier to overfit

Bayesian PMF

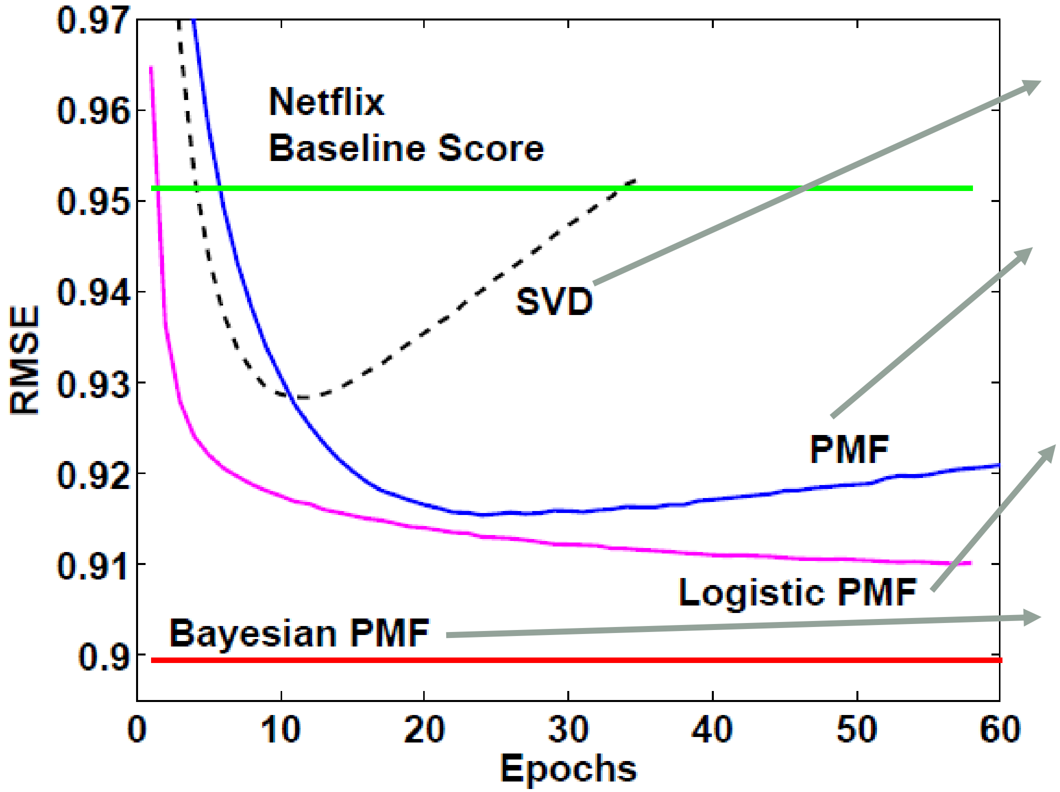
Use Bayesian inference to get the whole posterior distribution of U_i and V_j given the data R_{ij}

Covariances $\Lambda_U^{-1}, \Lambda_V^{-1}$ are learnable

Harder to overfit and better performance

Bayesian Probabilistic Matrix Factorization (BPMF): Experimental Results

Dataset: Netflix.
Size: 100M ratings, 480K users, 17K movies.



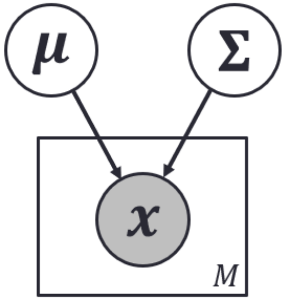
SVD: Very easy to overfit

PMF: Better performance and harder to overfit

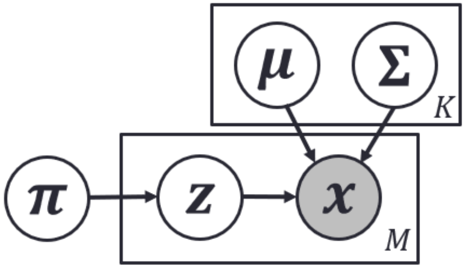
Logistic PMF: Even better performance

Bayes PMF: Best performance

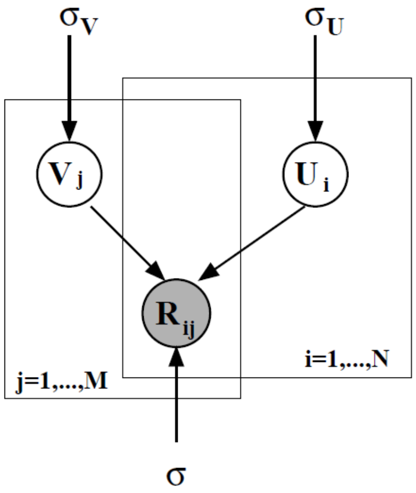
Summary on Probabilistic Graphical Models



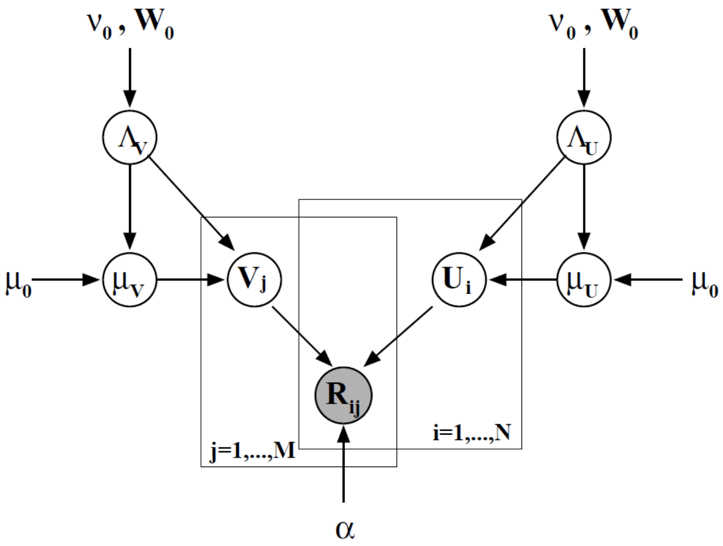
Gaussian Distribution



Mixture of Gaussians

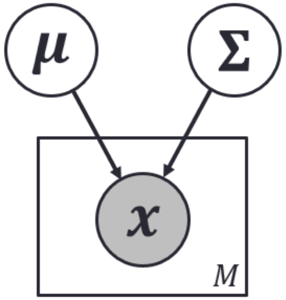


PMF

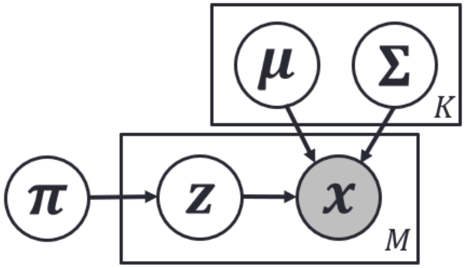


Bayesian PMF

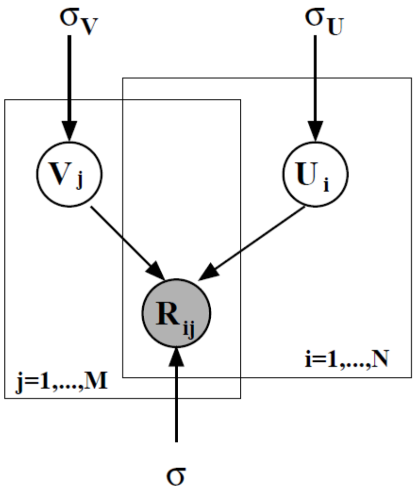
Summary on Learning and Inference Algorithms



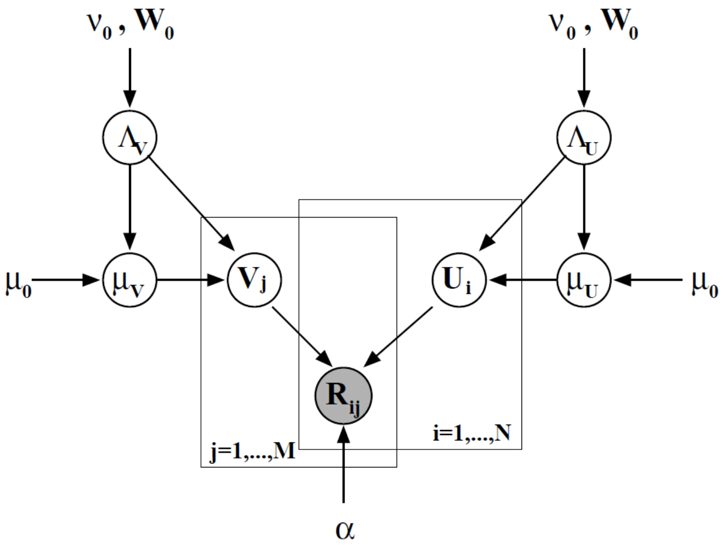
Maximum Likelihood Estimation (MLE)



EM



MAP



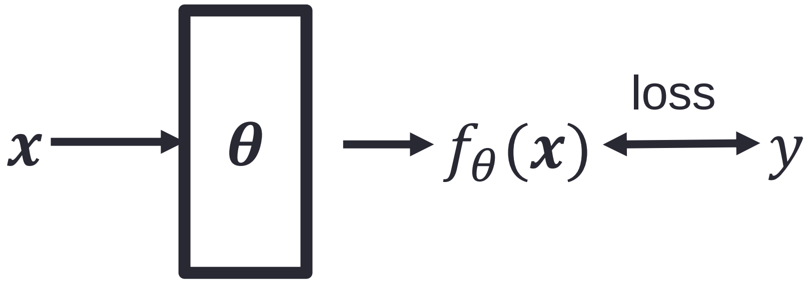
Gibbs Sampling

Bayesian Deep Learning

A Unified Framework

- Probabilistic Graphical Models
- Probabilistic/Bayesian Neural Nets

Neural Networks



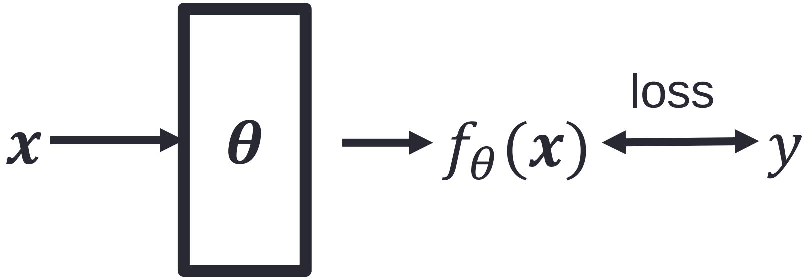
$$L_i = (f_\theta(\boldsymbol{x_i}) - y)^{\boldsymbol{2}}$$

$$L = \Sigma_{\mathrm{i}=1}^{N} \frac{1}{N} (f_\theta\left(\boldsymbol{x_i}\ \right) - y_i\ )^{\boldsymbol{2}}$$

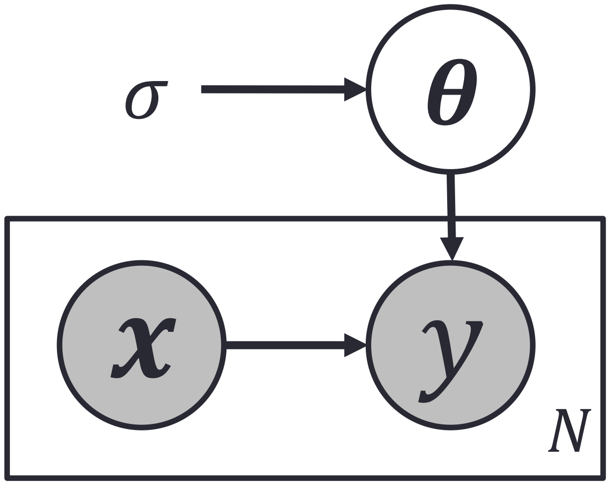
```
For each iteration $t = 1{:}T$ do
    Sample a minibatch of $n$ data points $(x, y)$
    Update parameters using stochastic gradient descent (SGD):
        $\theta_{t+1} = \theta_t - \Delta\theta_t$
        $\Delta\theta_t = \epsilon_t \left( \frac{1}{n} \sum_{i=1}^{n} \frac{\partial L_{ti}}{\partial \theta_t} \right)$
```

Bayesian Neural Networks

Neural Network



Bayesian Neural Network



Generative Process:

Generate $\theta \sim \mathrm{p}(\theta|\sigma)$

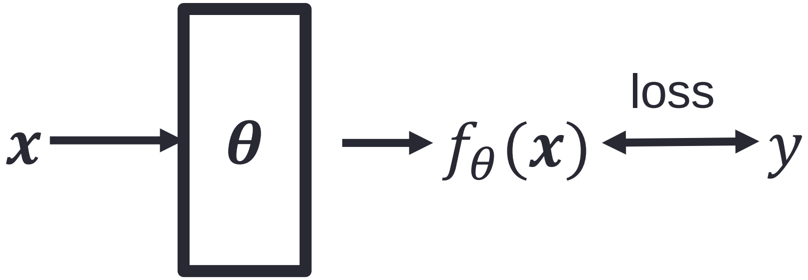
Generate $\mathrm{y} \sim \mathrm{p}(\mathrm{y}|\theta, \mathrm{x})$

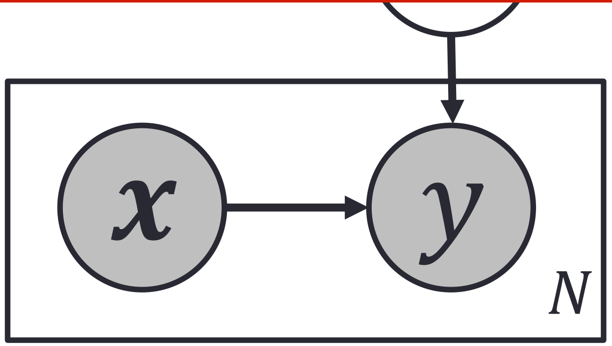
Bayesian Neural Networks

Neural Network

Bayesian Neural Network

How to learn the distribution of θ?





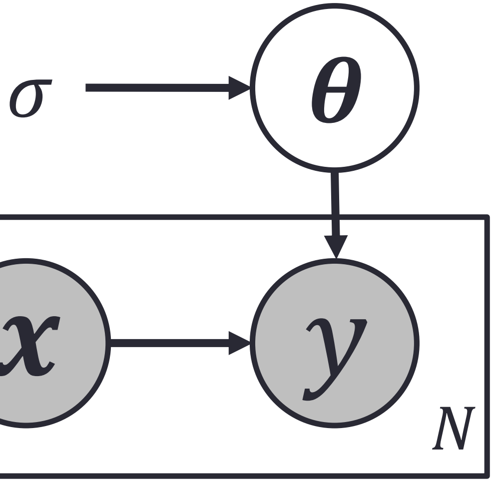
Learning: Given data points $\boldsymbol{x}_i, y_i$ and hyperparameter σ, estimate the distribution of neural network parameters θ, i.e., $p(\theta|x, y)$

Bayesian Neural Networks with Stochastic Gradient Langevin Dynamics (SGLD)

SGD:

$$\theta_{t+1} = \theta_t - \Delta\theta_t$$

$$\Delta\theta_t = \epsilon_t \left(\frac{1}{n} \sum\nolimits_{i=1}^{n} \frac{\partial L_{ti}}{\partial \theta_t} \right)$$



For each iteration $t = 1{:}T$ do

Sample a minibatch of n data points (x, y)

Update parameters using stochastic gradient Langevin dynamics (SGLD):

$$\theta_{t+1} = \theta_t - \Delta\theta_t$$

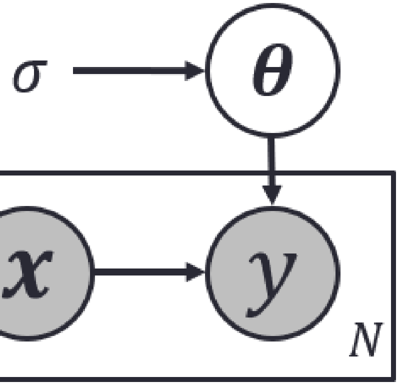
$$\Delta\theta_t = \frac{\epsilon_t}{2} \left(\frac{\partial ||\theta||_2^2}{\partial \theta} + \frac{N}{n} \sum\nolimits_{i=1}^{n} \frac{\partial L_{ti}}{\partial \theta} \right) + \eta_t$$

$$\eta_t \sim N(0, \epsilon_t)$$

Bayesian Neural Networks with SGLD

$$\theta_{t+1} = \theta_t - \Delta\theta_t$$

$$\Delta\theta_t = \epsilon_t \left(\frac{1}{n} \sum_{i=1}^{n} \frac{\partial L_{ti}}{\partial \theta_t} \right)$$



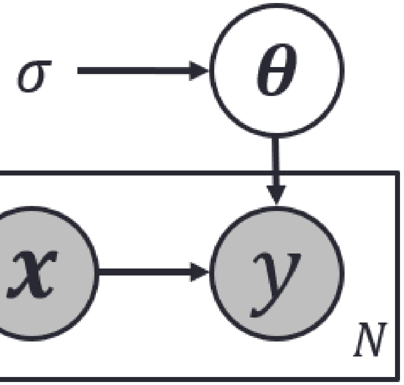
```
For each iteration $t = 1{:}T$ do
    Sample a minibatch of $n$ data points $(x, y)$
    Update parameters using SGLD:
        $\theta_{t+1} = \theta_t - \Delta\theta_t$
        $\Delta\theta_t = \frac{\epsilon_t}{2} \left( \frac{\partial \left|\left|\theta\right|\right|_2^2}{\partial \theta} + \frac{N}{n} \sum_{i=1}^{n} \frac{\partial L_{ti}}{\partial \theta} \right) + \boxed{\eta_t}$
        $\eta_t \sim N(0, \epsilon_t)$
```

Gaussian noise with the **variance** equal to **learning rate**

Bayesian Neural Networks with SGLD

$$\theta_{t+1} = \theta_t - \Delta\theta_t$$

$$\Delta\theta_t = \epsilon_t \left(\frac{1}{n} \sum_{i=1}^{n} \frac{\partial L_{ti}}{\partial \theta_t} \right)$$



```
For each iteration $t = 1:T$ do
    Sample a minibatch of $n$ data points $(x, y)$
    Update parameters using SGLD:
```

$$\theta_{t+1} = \theta_t - \Delta\theta_t$$

$$\Delta\theta_t = \frac{\epsilon_t}{2} \left(\boxed{\frac{\partial ||\theta||_2^2}{\partial \theta}} + \frac{N}{n} \sum_{i=1}^{n} \frac{\partial L_{ti}}{\partial \theta} \right) + \eta_t$$

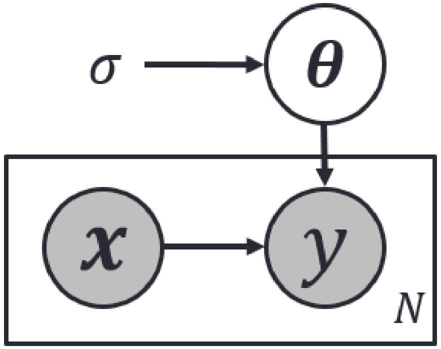
$$\eta_t \sim N(0, \epsilon_t)$$

L2 regularization term

Bayesian Neural Networks with SGLD

$$\theta_{t+1} = \theta_t - \Delta\theta_t$$

$$\Delta\theta_t = \epsilon_t \left(\frac{1}{n} \sum_{i=1}^{n} \frac{\partial L_{ti}}{\partial \theta_t} \right)$$



For each iteration $t = 1 : T$ do

Sample a minibatch of n data points (x, y)

Update parameters using stochastic gradient descent:

$$\theta_{t+1} = \theta_t - \Delta\theta_t$$

$$\Delta\theta_t = \frac{\epsilon_t}{2} \left(\frac{\partial ||\theta||_2^2}{\partial \theta} + \frac{N}{n} \sum_{i=1}^{n} \frac{\partial L_{ti}}{\partial \theta} \right) + \eta_t$$

$$\eta_t \sim N(0, \epsilon_t)$$

After convergence, sampling from $\theta_{t+1} = \theta_t - \Delta\theta_t$ is equivalent to sampling from true posterior distribution of NN parameters $p(\theta|x, y)$

Bayesian Neural Networks with SGLD

Generative Process:

Generate $\theta \sim \mathrm{p}(\theta|\sigma)$

Generate $\mathrm{y} \sim \mathrm{p}(\mathrm{y}|\theta, \mathrm{x})$

$$\Delta\theta_t = \frac{\epsilon_t}{2}\left(\frac{\partial||\theta||_2^2}{\partial\theta} + \frac{N}{n}\sum\nolimits_{i=1}^{n}\frac{\partial L_{ti}}{\partial\theta}\right) + \eta_t$$

$$\Delta\theta_t = -\frac{\epsilon_t}{2}\left(\frac{\partial \log p(\theta|\sigma)}{\partial\theta} + \frac{N}{n}\sum\nolimits_{i=1}^{n}\frac{\partial \log p(y|x_{ti},\theta)}{\partial\theta}\right) + \eta_t$$

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \quad \Rightarrow \quad \mu = 0, \sigma = 1 \rightarrow \log f(x) = -\frac{1}{2}x^2 + C$$

After convergence, sampling from $\theta_{t+1} = \theta_t - \Delta\theta_t$ is equivalent to sampling from true posterior distribution of NN parameters $p(\theta|x,y)$

$$\log p(\theta|x,y,\sigma) \;= \log p(\theta|\sigma) + \log p(y|x,\theta,\sigma) + C$$

Bayesian Learning via Stochastic Gradient Langevin Dynamics, Welling and Teh, ICML 2011

Bayesian Neural Networks with SGLD: Experimental Results

UCI adult dataset;
32561 observations and 123 features;
classification task.

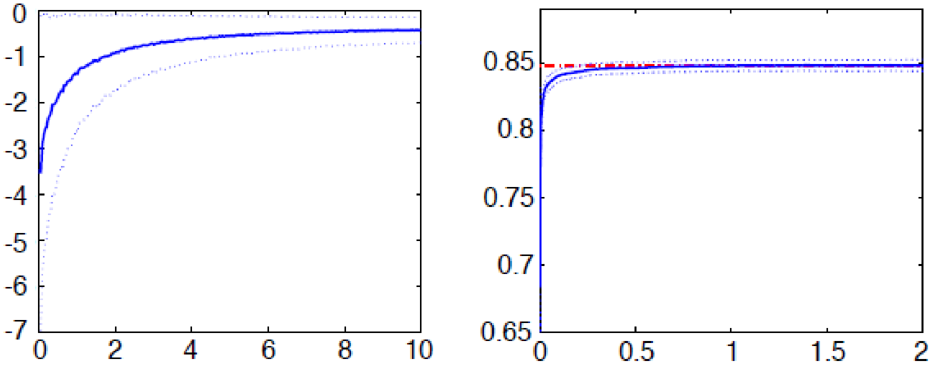


Figure 3. Average log joint probability per data item (left) and accuracy on test set (right) as functions of the number of sweeps through the whole dataset. Red dashed line represents accuracy after 10 iterations. Results are averaged over 50 runs; blue dotted lines indicate 1 standard deviation.

Bayesian Learning via Stochastic Gradient Langevin Dynamics, Welling and Teh, ICML 2011

Bayesian Neural Networks with SGLD: Experimental Results

MNIST dataset;
60K observations and 784 features;
classification task.

Table: Test set misclassification rate on MNIST for different methods using a 784-400-400-10 MLP.

SGD	Dropout	SGLD
1.83	1.51	1.27

Bayesian Learning via Stochastic Gradient Langevin Dynamics, Welling and Teh, ICML 2011
Bayesian Dark Knowledge, Korattikara et al., NIPS 2015

Bayesian Neural Networks with SGLD: Price to Pay

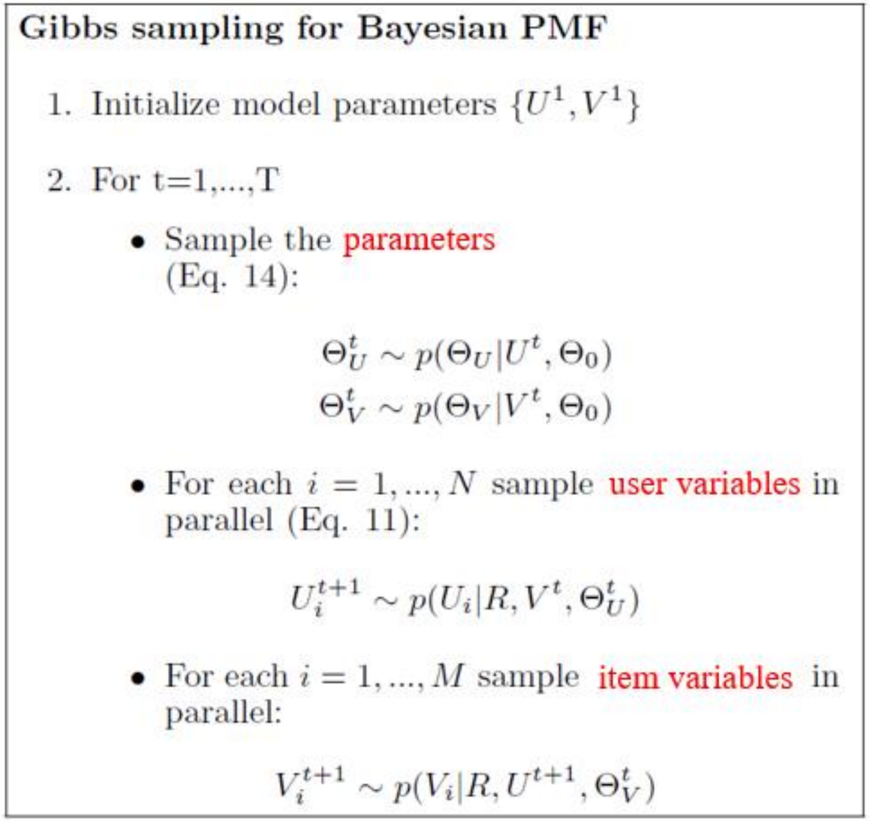
1. Storage and Memory

   Store multiple copies of neural network parameters

2. Computation Time

   Multiple passes of feedforward inferences $f(x|\theta_t)$

After Learning Bayesian PMF, How to Make Predictions (Recap)



Gibbs sampling for Bayesian PMF

```
1. Initialize model parameters $\{U^1, V^1\}$
2. For t=1,...,T
    • Sample the parameters (Eq. 14):
        $\Theta_U^t \sim p(\Theta_U | U^t, \Theta_0)$
        $\Theta_V^t \sim p(\Theta_V | V^t, \Theta_0)$
    • For each $i = 1, ..., N$ sample user variables in parallel (Eq. 11):
        $U_i^{t+1} \sim p(U_i | R, V^t, \Theta_U^t)$
    • For each $i = 1, ..., M$ sample item variables in parallel:
        $V_i^{t+1} \sim p(V_i | R, U^{t+1}, \Theta_V^t)$
```

$$p(R_{ij}^* | R, \Theta_0) \approx \frac{1}{K} \sum_{k=1}^{K} p(R_{ij}^* | U_i^{(k)}, V_j^{(k)})$$

These K samples $U_i^{(k)}, V_j^{(k)}$ are generated by running K additional iterations after the Gibbs sampling algorithm converges

Bayesian Neural Networks with SGLD: Price to Pay

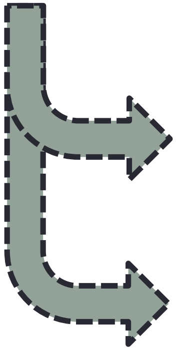
1. Store multiple copies of neural network parameters
2. Multiple passes of feedforward inferences $f(x|\theta_t)$

$$E[f(x|\theta_t)] \approx \frac{1}{T}\sum_{t=1}^{T} f(x|\theta_t)$$

Need T times the storage/memory cost and computation cost

Bayesian Dark Knowledge, Korattikara et al., NIPS 2015

Bayesian Deep Learning

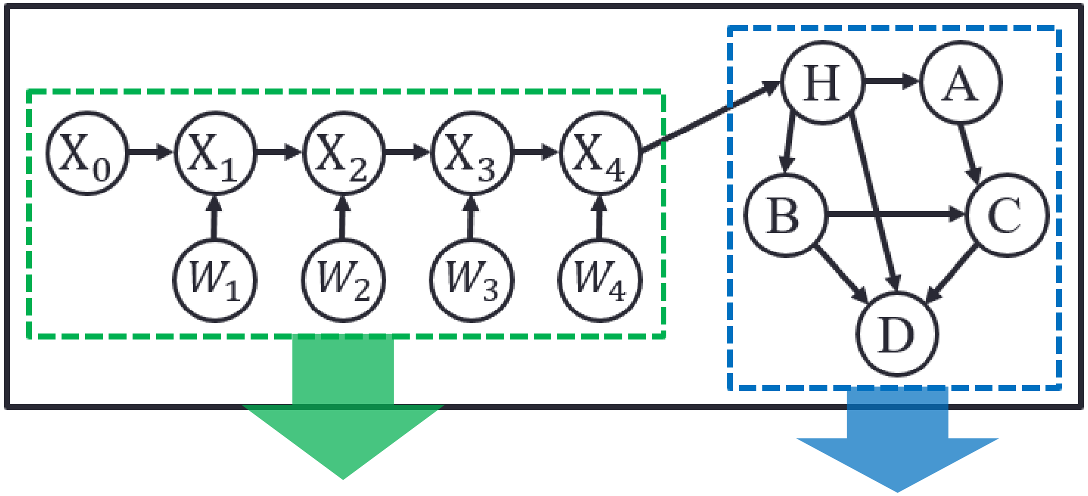


A Unified Framework

Probabilistic Graphical Models

Probabilistic/Bayesian Neural Nets

BDL: A Principled Probabilistic Framework (Recap)



Deep Component

Graphical Component

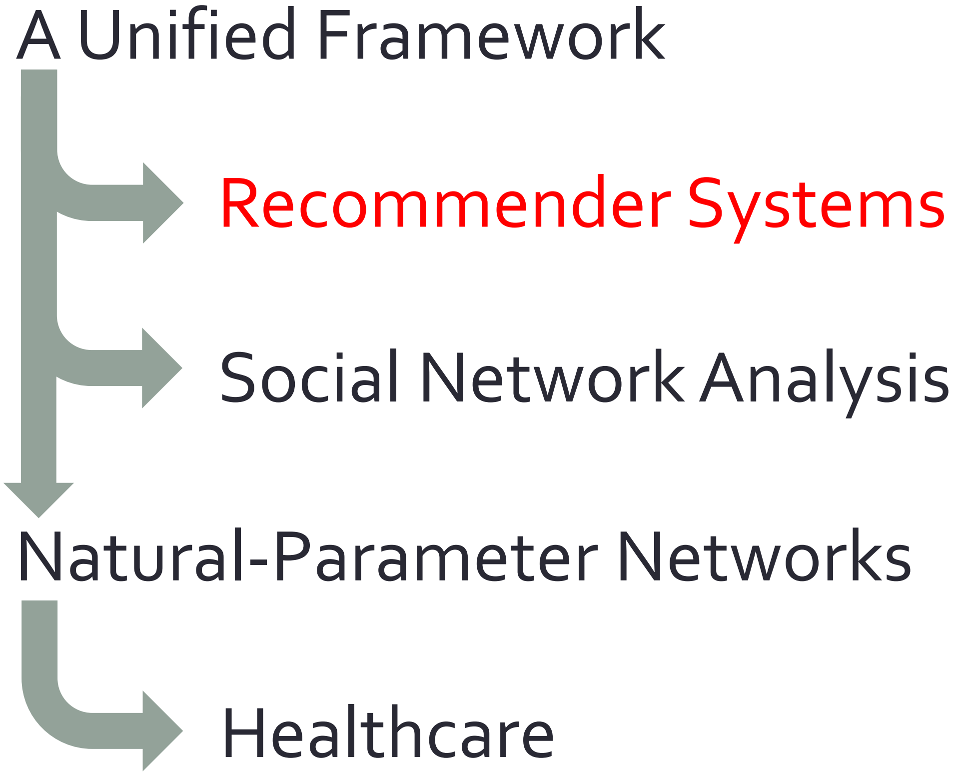
Deep Variables X_n W_n

Graphical Variables A B C D

Hinge Variables H

[**Wang** et al. TKDE 2016]

Bayesian Deep Learning



A Unified Framework

Recommender Systems

Social Network Analysis

Natural-Parameter Networks

Healthcare

[**Wang** et al., KDD 2015]
[**Wang** et al., NIPS 2016a]

Recommender Systems

Rating matrix:

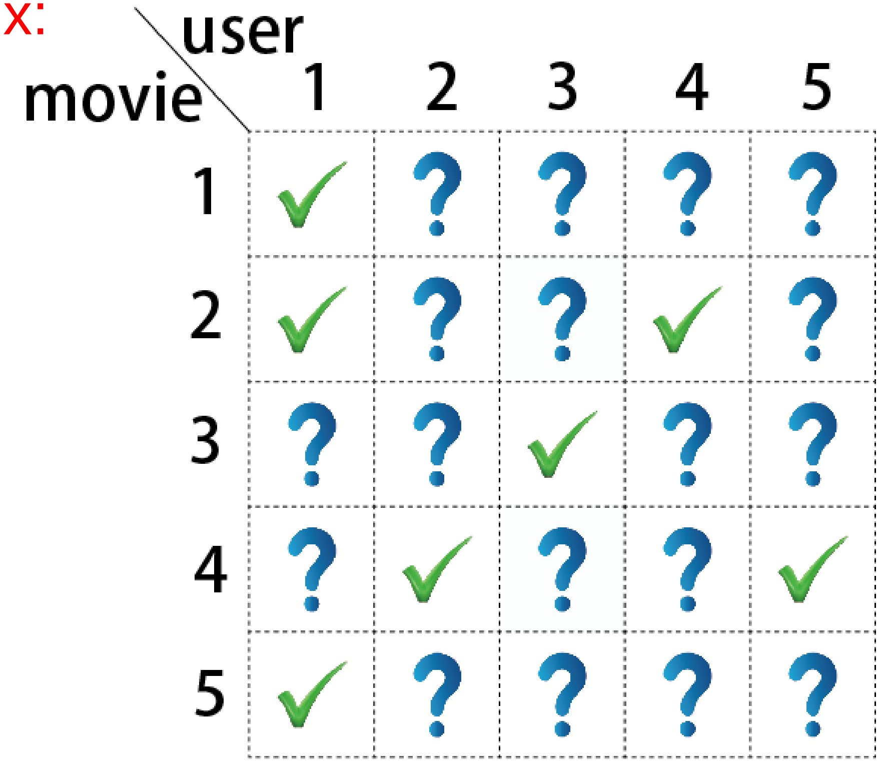
movie \ user	1	2	3	4	5
1	✓				
2	✓			✓	
3			✓		
4		✓			✓
5	✓				

Observed preferences: ✓

Matrix completion

Recommender Systems

Rating matrix:



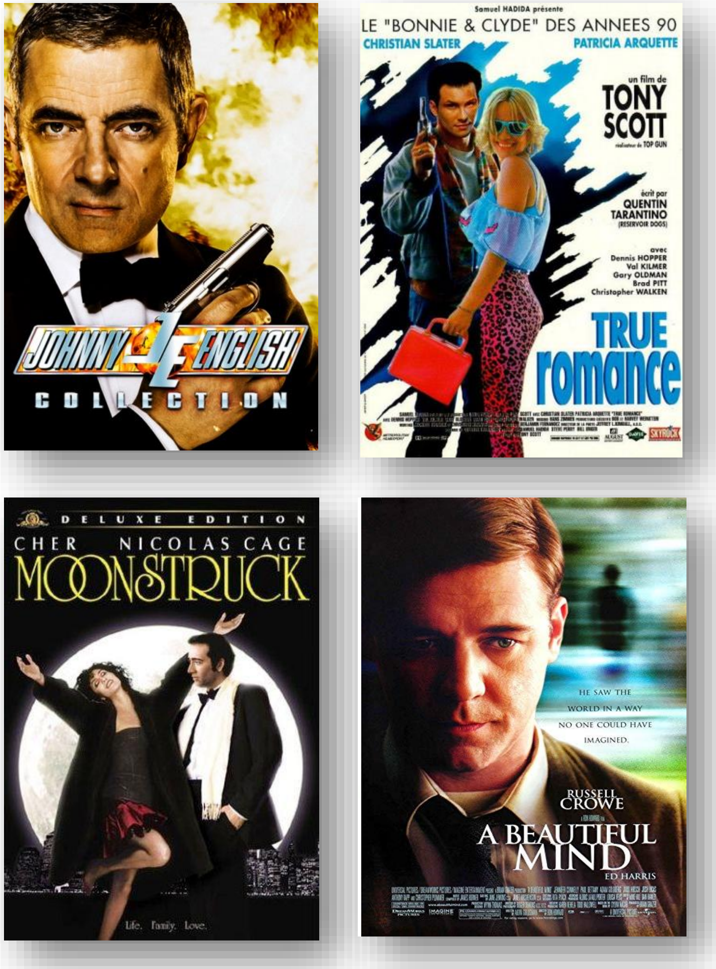
movie \ user	1	2	3	4	5
1	✓	?	?	?	?
2	✓	?	?	✓	?
3	?	?	✓	?	?
4	?	✓	?	?	✓
5	✓	?	?	?	?

Matrix completion ⇒

Observed preferences: ✓

To predict: ?

Recommender Systems with Content



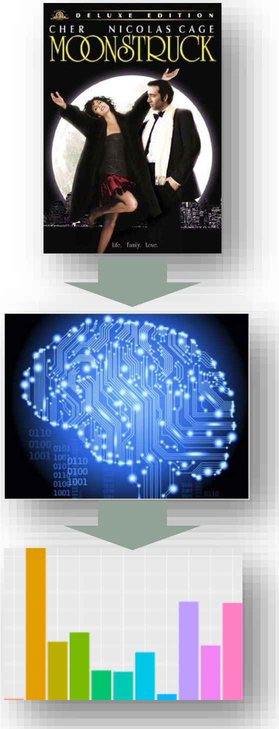
movie \ user	1	2	3	4	5
1	✓	?	?	?	?
2	✓	?	?	✓	?
3	?	?	✓	?	?
4	?	✓	?	?	✓
5	✓	?	?	?	?

Content information:
Plots, directors, actors, etc.

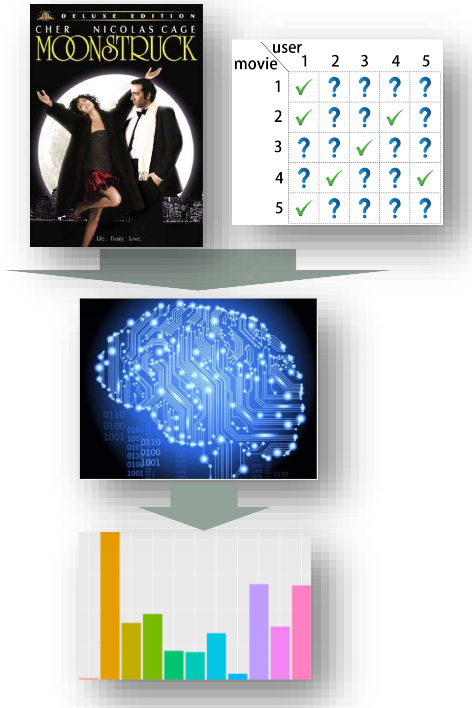
Sparse rating matrix

Modeling the Content Information

Handcrafted features



Automatically learn features



user movie	1	2	3	4	5
1	✓	?	?	?	?
2	✓	?	?	✓	?
3	?	?	✓	?	?
4	?	✓	?	?	✓
5	✓	?	?	?	?

Automatically learn features and **adapt for ratings**

Prior work

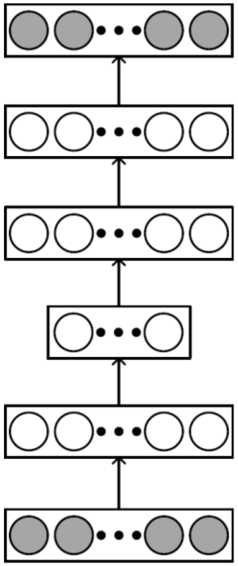
Our work

Modeling the Content Information

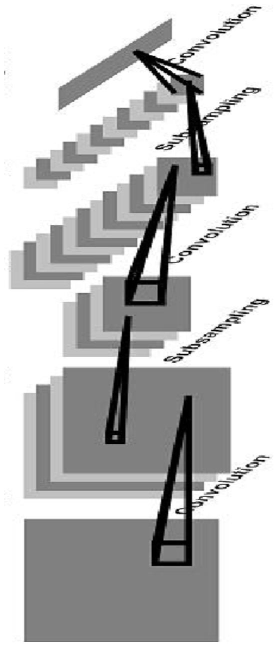
1. Powerful features for content information

Deep learning

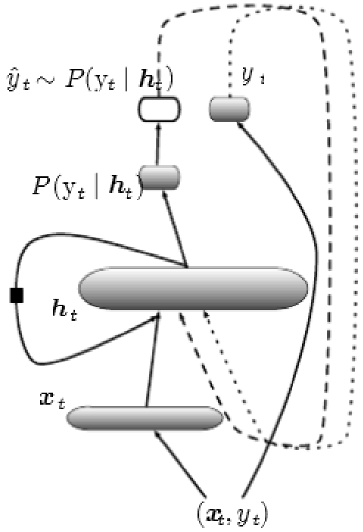
Deep Learning



Stacked denoising autoencoders



Convolutional neural networks



Recurrent neural networks

Typically for independent data points

i.e., no correlation between users and items

Modeling the Content Information

1. Powerful features for content information

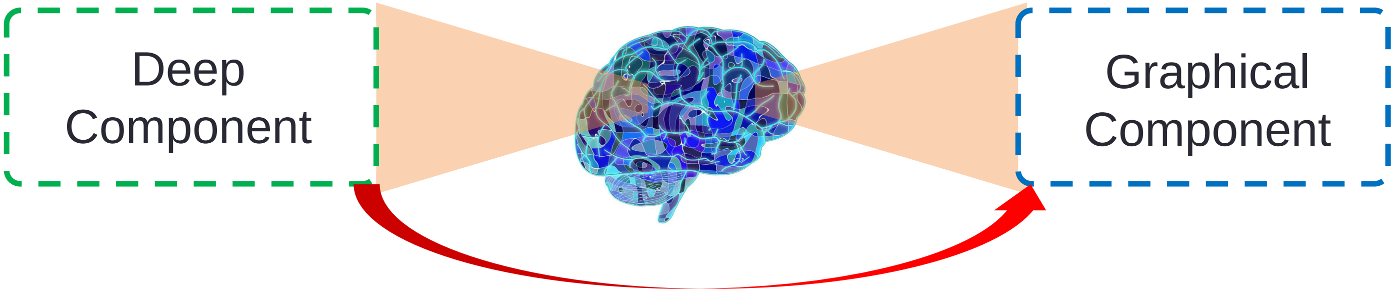
Deep learning

2. Feedback from rating information

Non-independent

Collaborative deep learning (CDL)

Challenges



Deep Component

Graphical Component

1. Probabilistic deep learning models as a **deep component**

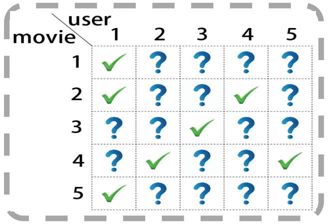
   Compatible with the graphical component

   Powerful as non-probabilistic versions

2. **Connect** to the graphical component

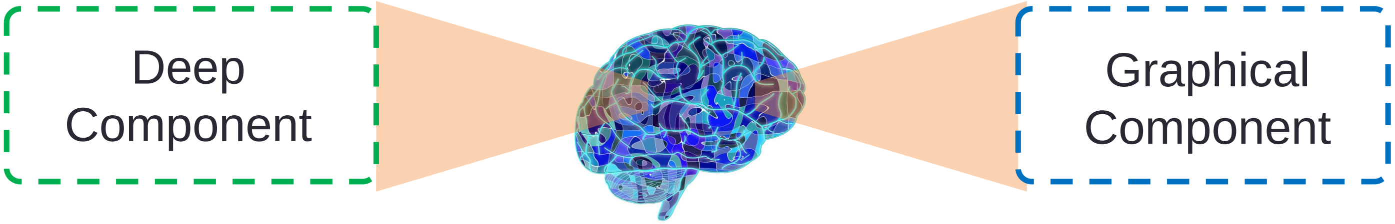
   Similarity, preferences

   Recommendation



user movie	1	2	3	4	5
1	✓	?	?	?	?
2	✓	?	?	✓	?
3	?	?	✓	?	?
4	?	✓	?	?	✓
5	✓	?	?	?	?

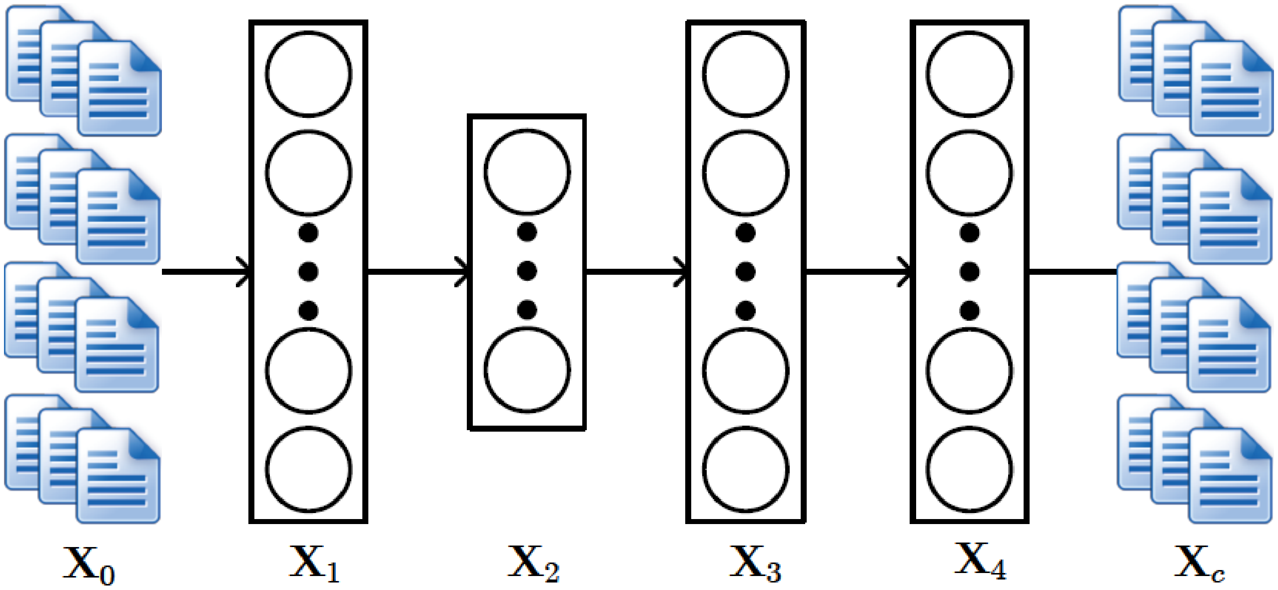
Challenge 1



1. Probabilistic deep learning models as a **deep component**
    - **Compatible** with the graphical component
    - **Powerful** as non-probabilistic versions

2. **Connect** to the graphical component
    - Similarity, preferences
    - Recommendation

Challenge 1
Step 1 of 2: Autoencoder (AE)



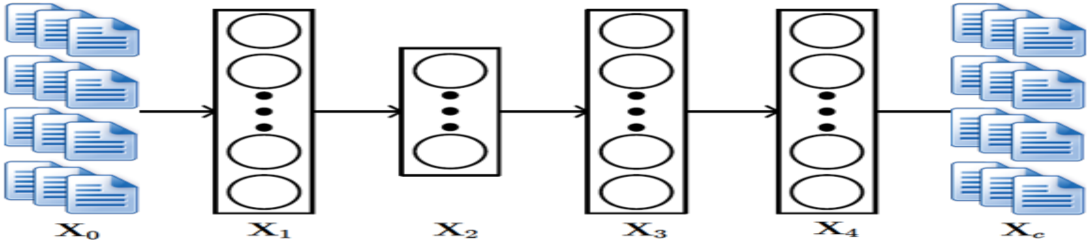
Content
(e.g., documents)

Content
(e.g., documents)

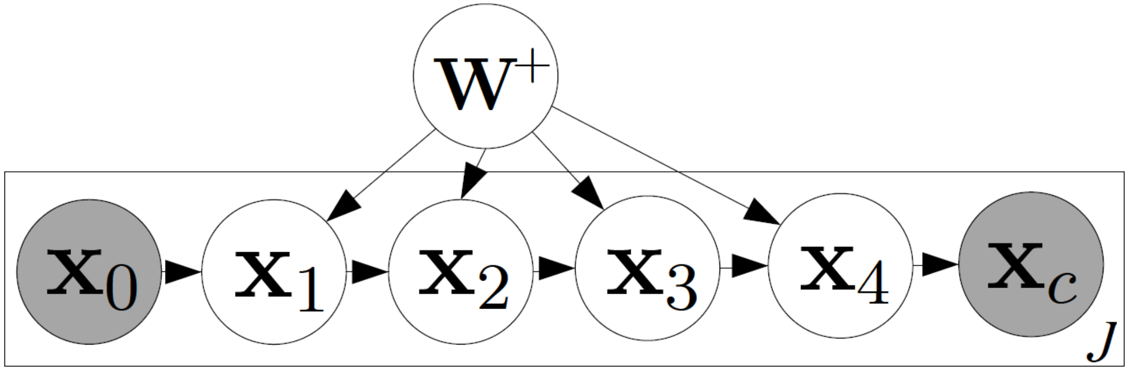
X_2: Middle-Layer representation

Challenge 1
Step 2 of 2: Probabilistic Autoencoder

Standard autoencoder:



Probabilistic autoencoder: (ours)

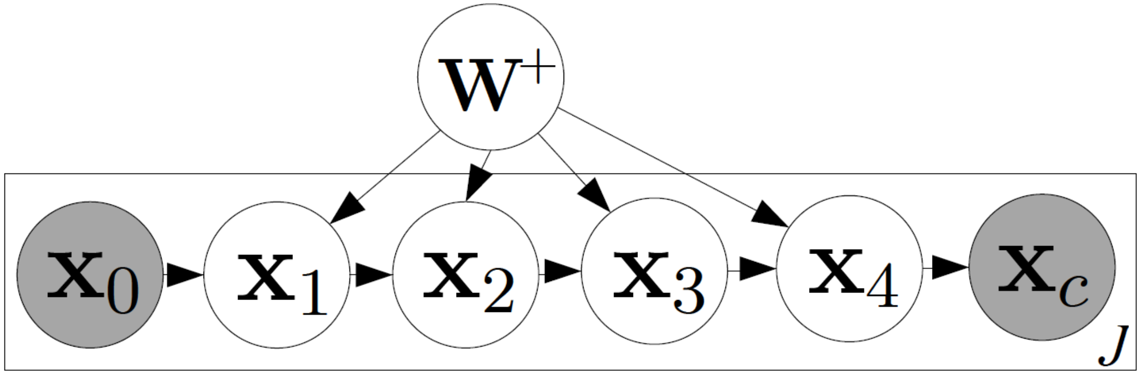


$$\mathbf{X}_{l,j*} \sim \mathcal{N}(\sigma(\mathbf{X}_{l-1,j*}\mathbf{W}_l + \mathbf{b}_l), \boxed{\lambda_s^{-1}\mathbf{I}_{K_l}})$$

Probabilistic Autoencoder: Gaussian noise after each nonlinear transformation

[**Wang** et al. KDD 2015]

Challenge 1
Step 2 of 2: Probabilistic Autoencoder

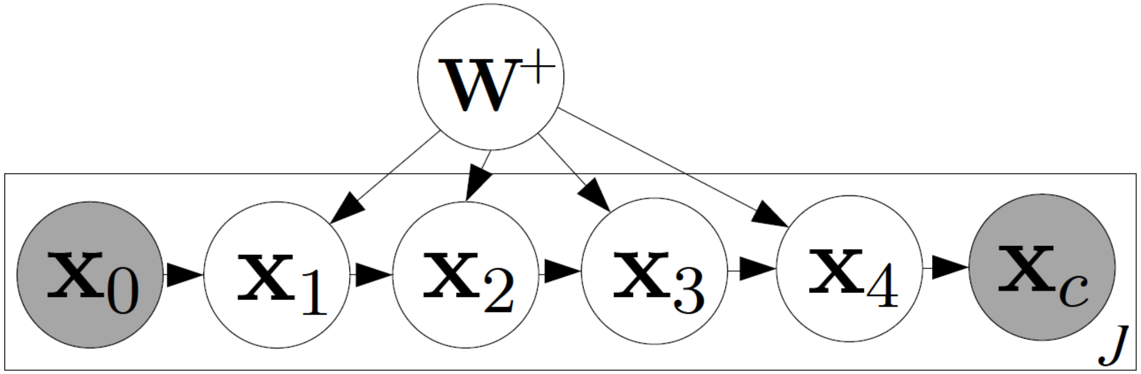


Probabilistic Autoencoder

X_2: **Middle-Layer representation**

[**Wang** et al. KDD 2015]

Challenge 1
Step 2 of 2: Probabilistic Autoencoder



Probabilistic Autoencoder

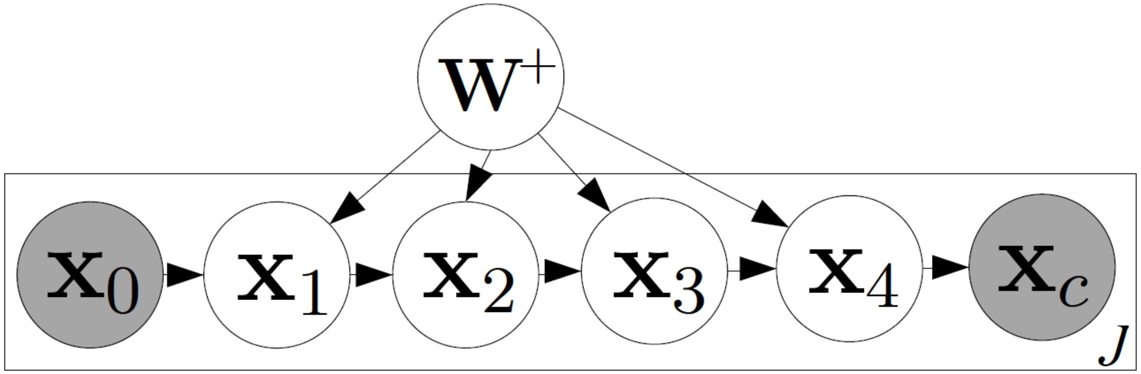
X_2: **Middle-Layer representation**

● Observed variables (**given**)

[**Wang** et al. KDD 2015]

Challenge 1

Step 2 of 2: Probabilistic Autoencoder



Probabilistic Autoencoder

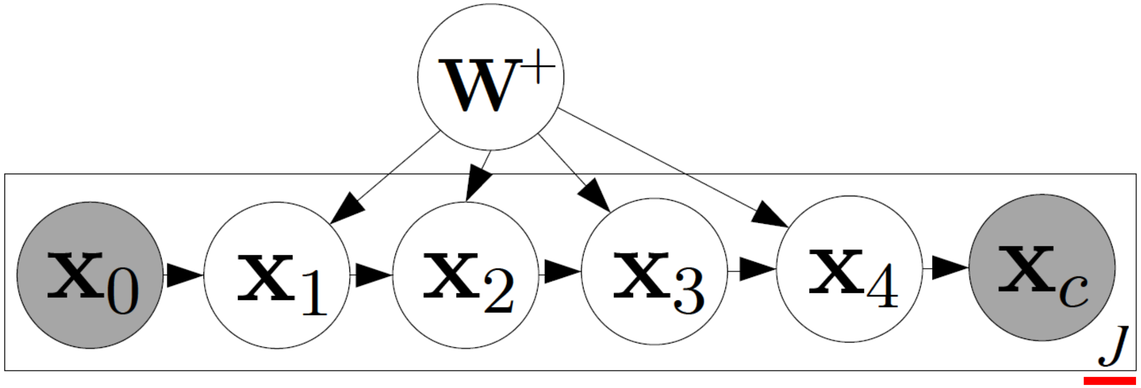
X_2: **Middle-Layer representation**

● Observed variables (**given**)

○ Latent variables & parameters **to learn**

[**Wang** et al. KDD 2015]

Challenge 1
Step 2 of 2: Probabilistic Autoencoder



Probabilistic Autoencoder

X_2: **Middle-Layer representation**

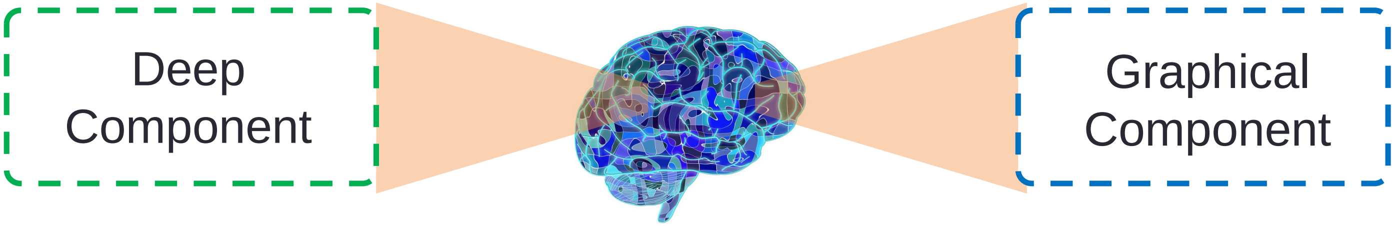
● Observed variables (**given**)

○ Latent variables & parameters **to learn**

J Number of documents

[**Wang** et al. KDD 2015]

Challenge 1



Deep Component

Graphical Component

1. Probabilistic deep learning models as a **deep component**

    Compatible with the graphical component

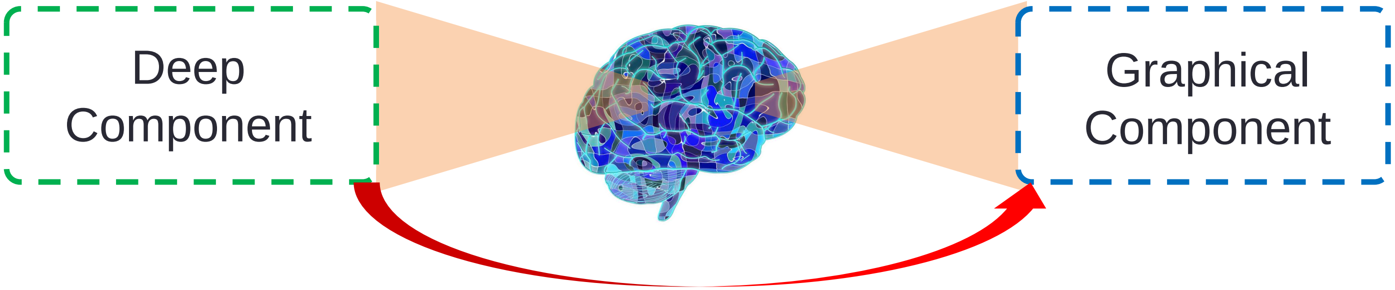
    Powerful as non-probabilistic versions

2. **Connect** to the graphical component

    Similarity, preferences

    Recommendation

Challenge 2

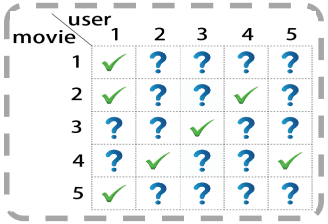


Deep Component

Graphical Component

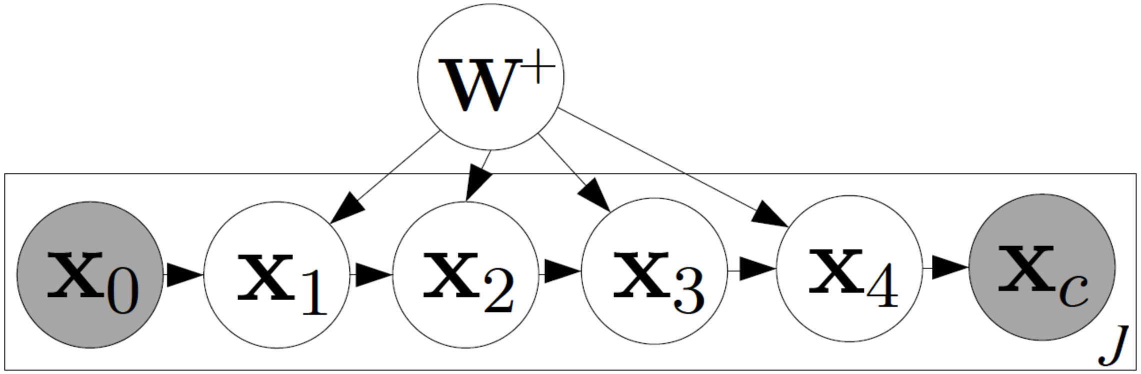
1. Probabilistic deep learning models as a **deep component**
   - **Compatible** with the graphical component
   - **Powerful** as non-probabilistic versions

2. **Connect** to the graphical component
   - Similarity, preferences
   - Recommendation



user movie	1	2	3	4	5
1	✓	?	?	?	?
2	✓	?	?	✓	?
3	?	?	✓	?	?
4	?	✓	?	?	✓
5	✓	?	?	?	?

Challenge 2
Step 1 of 4: Start from Middle-Layer Representation

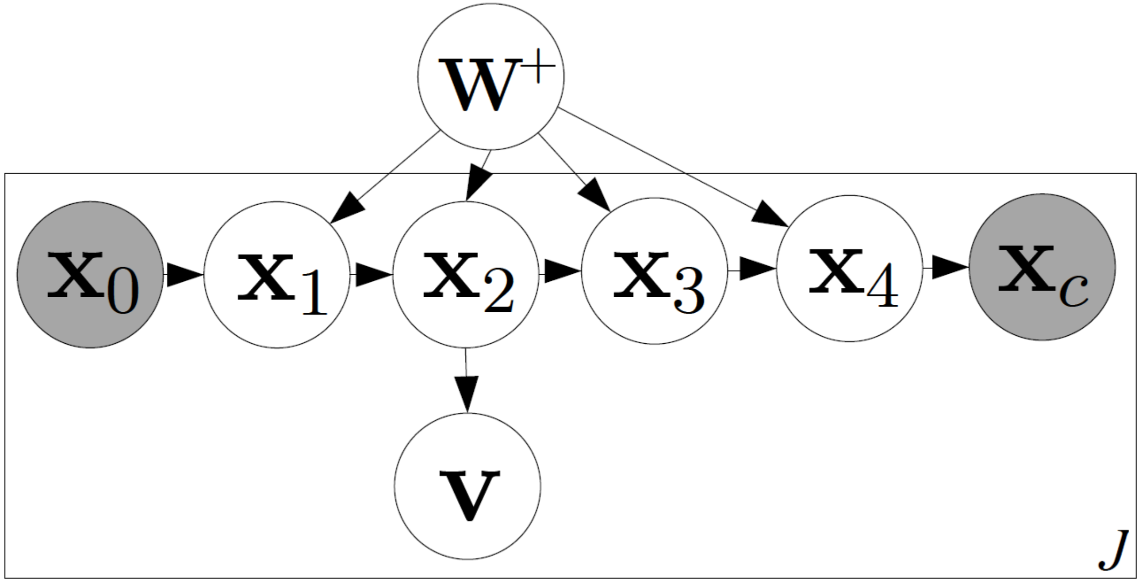


Start from probabilistic Autoencoder

X_2: Middle-Layer representation

[**Wang** et al. KDD 2015]

Challenge 2
Step 2 of 4: Generate Item j's Latent Vector v_j

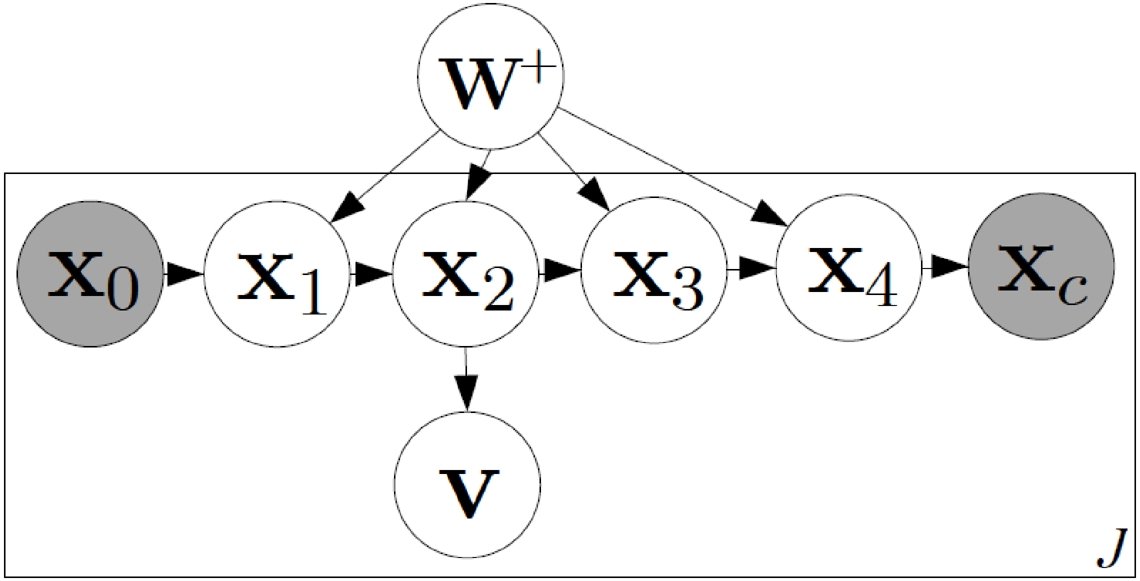


Generate the **latent vector for item j** from X_2:

$$\underline{\mathbf{v}_j} \sim \mathcal{N}(\mathbf{X}_2, \lambda_v^{-1}\mathbf{I})$$

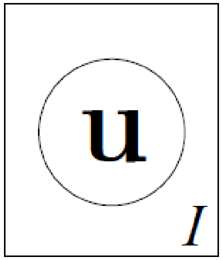
[**Wang** et al. KDD 2015]

Challenge 2
Step 3 of 4: Generate User i's Latent Vector u_i



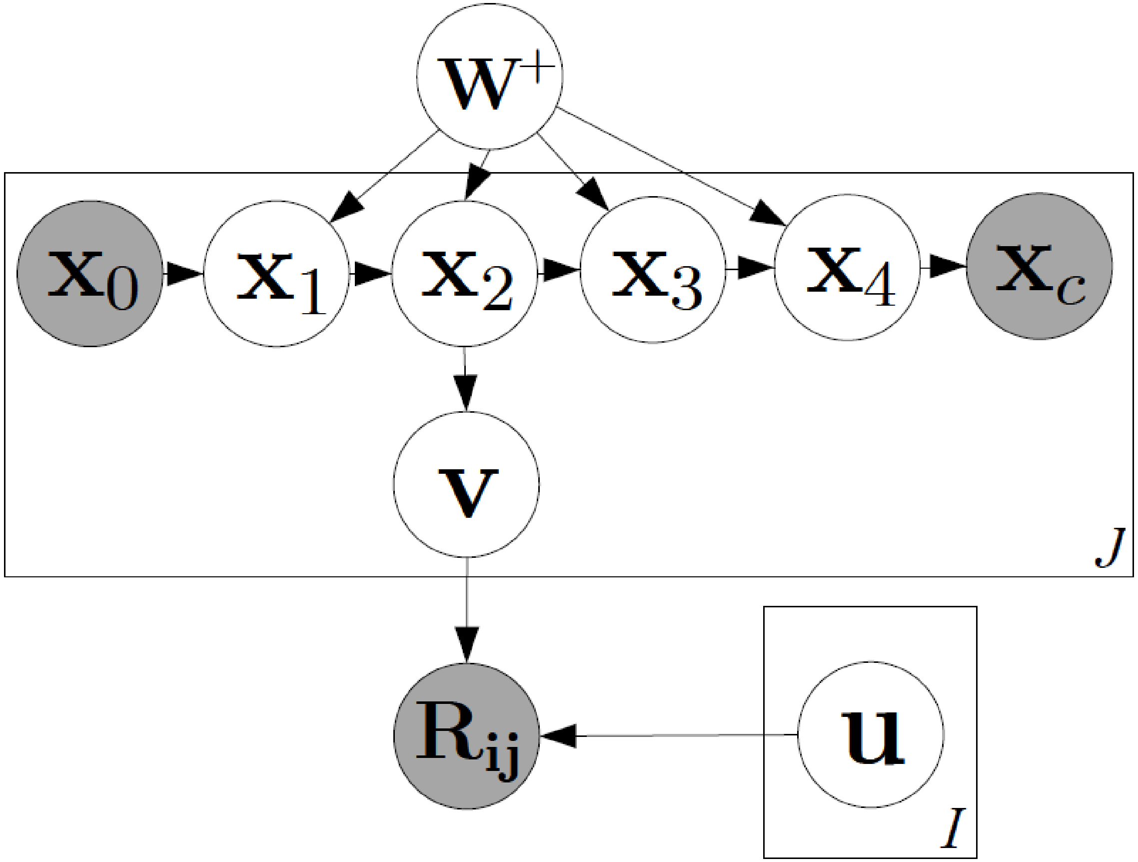
Generate the **latent vector for user i**:

$$\underline{\mathbf{u}_i} \sim \mathcal{N}(\mathbf{0}, \lambda_u^{-1}\mathbf{I})$$



[**Wang** et al. KDD 2015]

Challenge 2
Step 4 of 4: Generate Ratings R_{ij} from $u_i^T v_j$

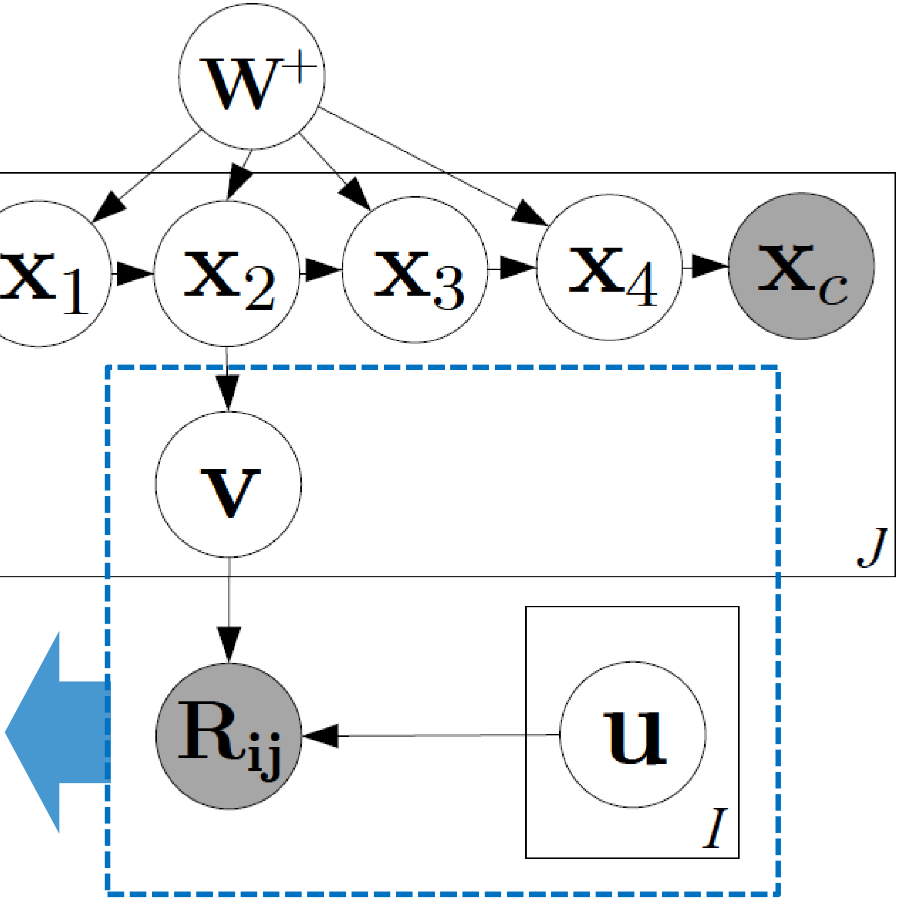


Generate the **rating user i gives item j**: $\underline{\mathbf{R}_{ij}} \sim \mathcal{N}(\underline{\mathbf{u}_i^T \mathbf{v}_j}, \lambda_r^{-1})$

[**Wang** et al. KDD 2015]

Challenge 2

Step 4 of 4: Generate Ratings R_{ij} from $u_i^T v_j$



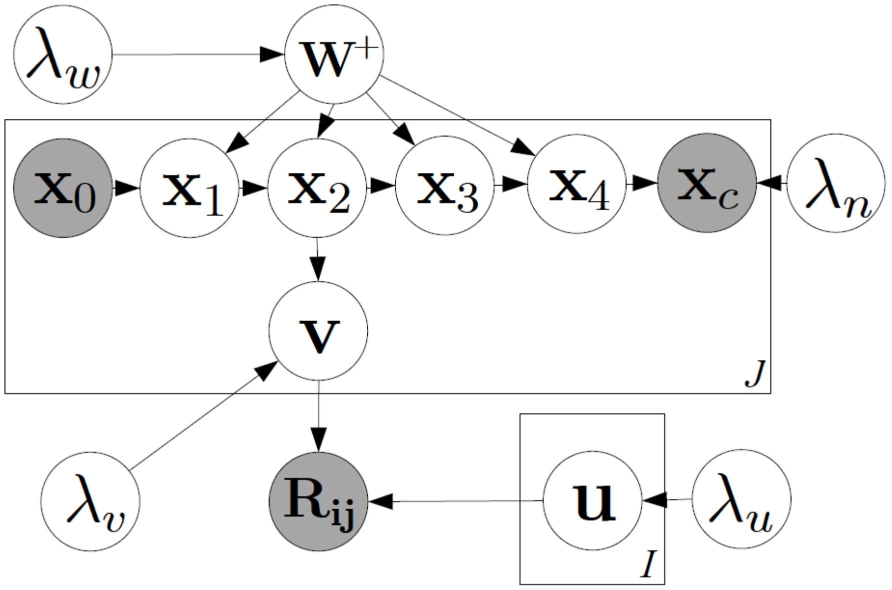
Graphical Component

Generate the **rating user i gives item j**: $\underline{\mathbf{R}_{ij}} \sim \mathcal{N}(\underline{\mathbf{u}_i^T \mathbf{v}_j}, \lambda_r^{-1})$

[**Wang** et al. KDD 2015]

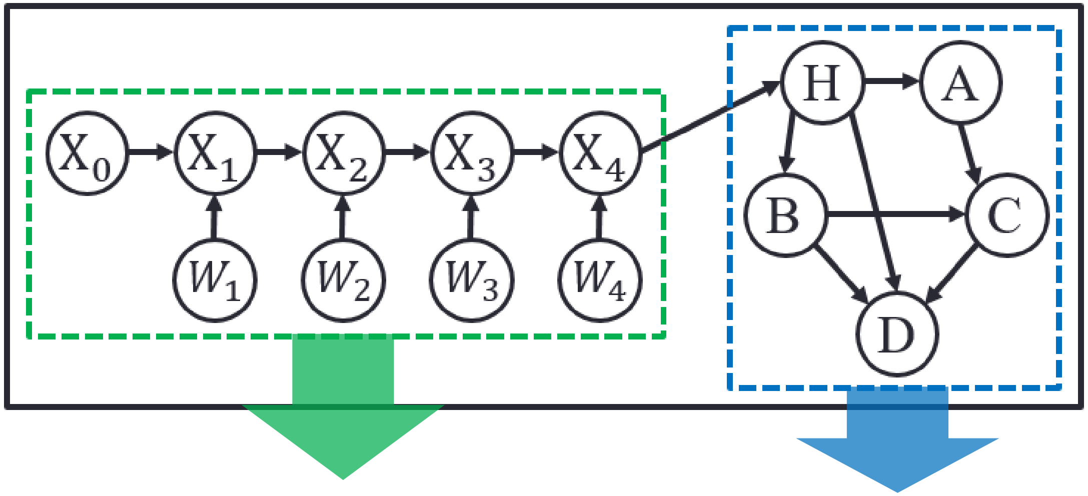
Overview: Collaborative Deep Learning (CDL)

Graphical model:



λ_w, λ_n, λ_v, λ_u:
hyperparamters to control the **variance** of Gaussian distributions

BDL: A Principled Probabilistic Framework (Recap)



Deep Component

Graphical Component

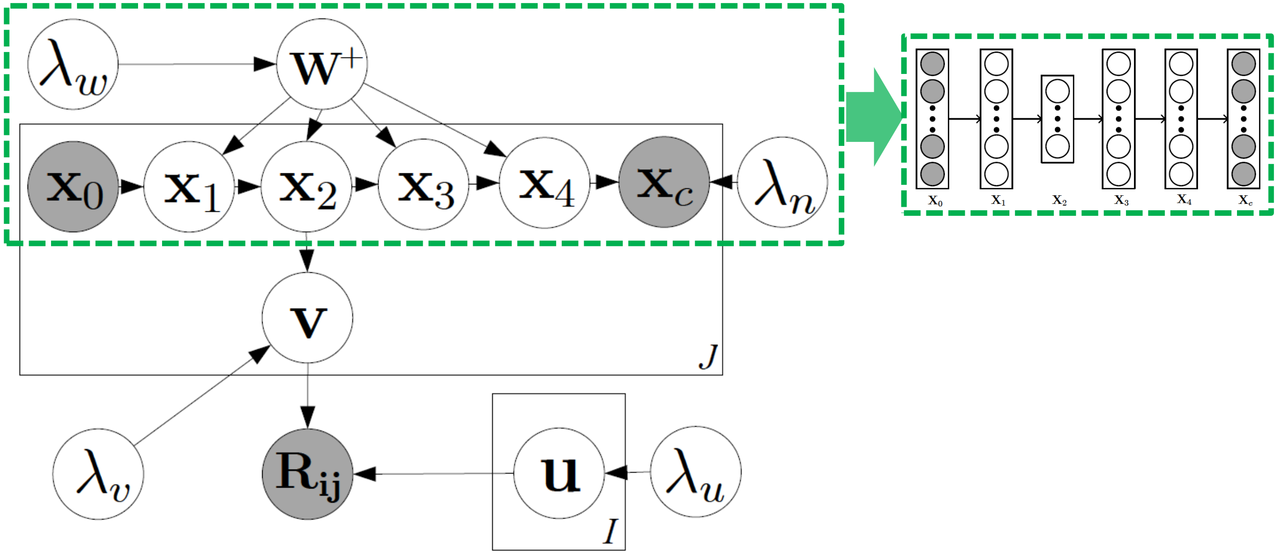
Deep Variables X_n W_n

Graphical Variables A B C D

Hinge Variables H

[**Wang** et al. TKDE 2016]

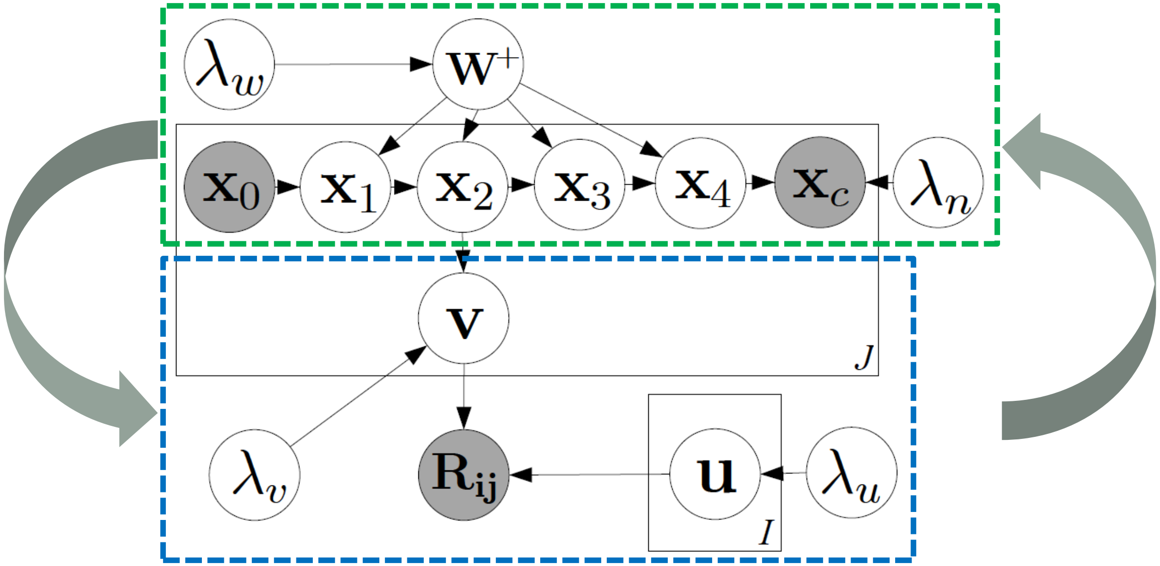
Graphical Model of CDL with Two Components



Collaborative deep learning

Graphical Model of CDL with Two Components

Boost recommendation accuracy



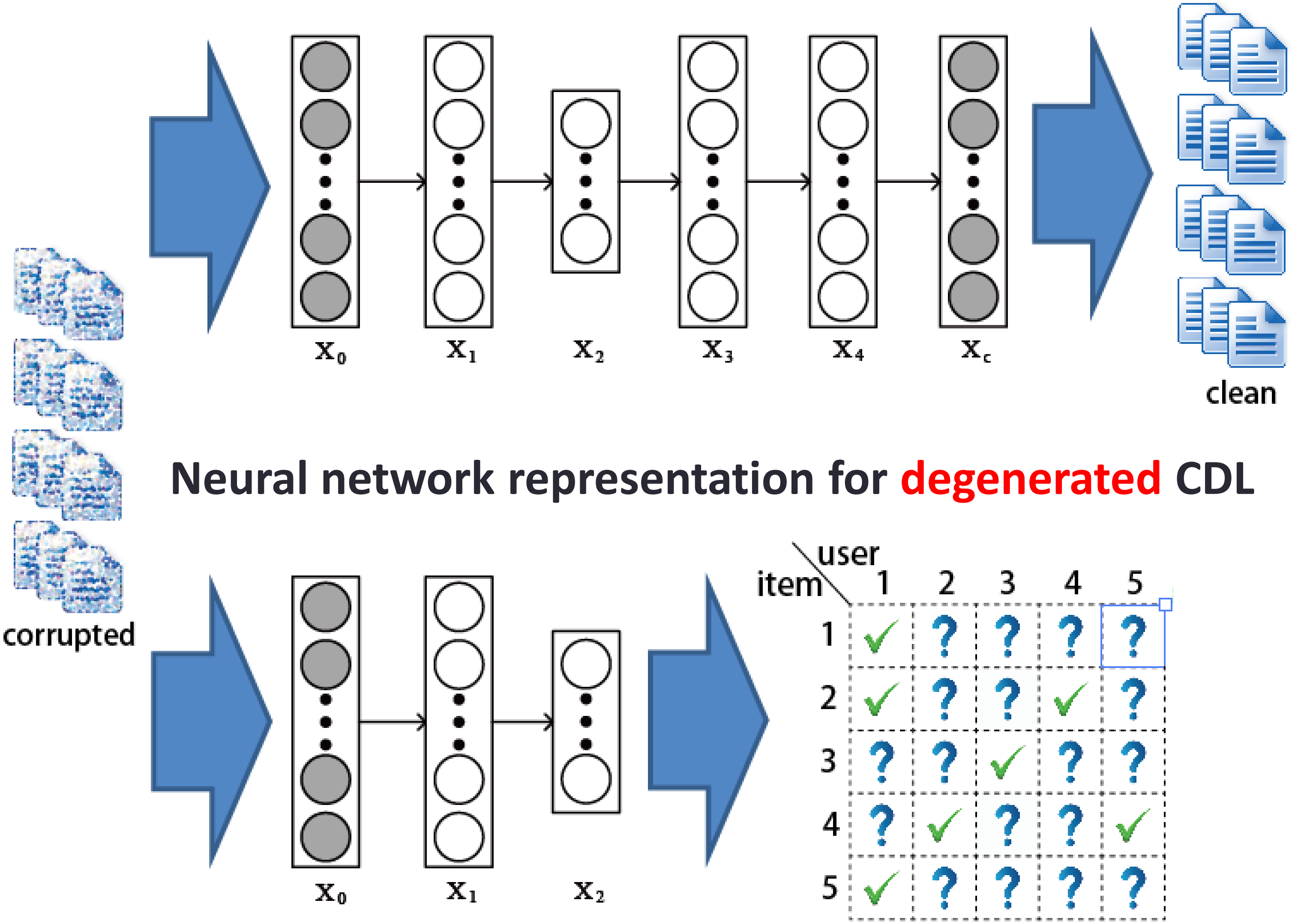
Boost representation learning

Collaborative deep learning

Trained end-to-end

- **Boost each other's performance**
- More powerful representation
- Infer missing ratings from content
- Infer missing content from ratings

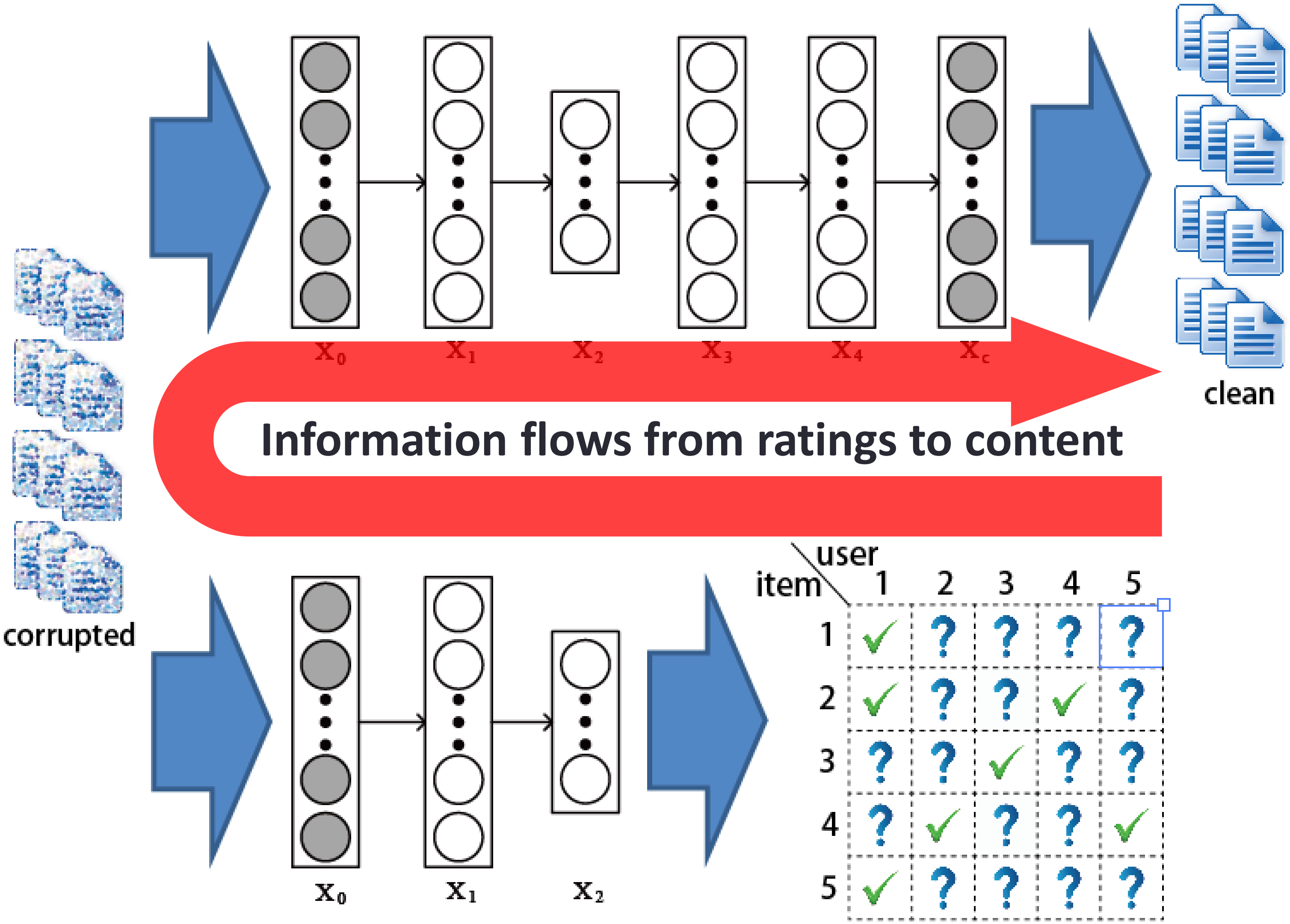
Collaborative Deep Learning



Neural network representation for **degenerated** CDL

item \ user	1	2	3	4	5
1	✓	?	?	?	?
2	✓	?	?	✓	?
3	?	?	✓	?	?
4	?	✓	?	?	✓
5	✓	?	?	?	?

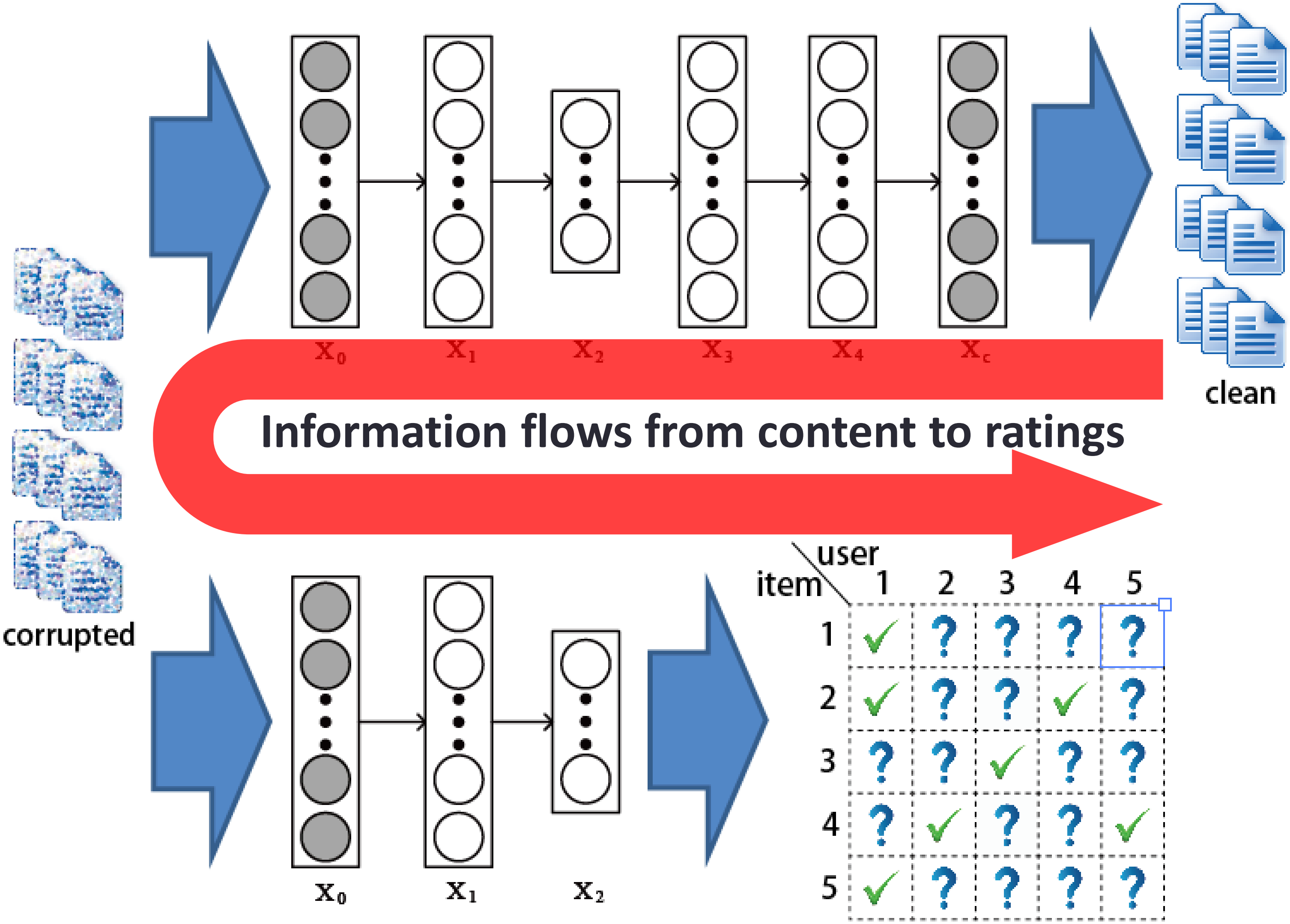
Collaborative Deep Learning



Information flows from ratings to content

item \ user	1	2	3	4	5
1	✓	?	?	?	?
2	✓	?	?	✓	?
3	?	?	✓	?	?
4	?	✓	?	?	✓
5	✓	?	?	?	?

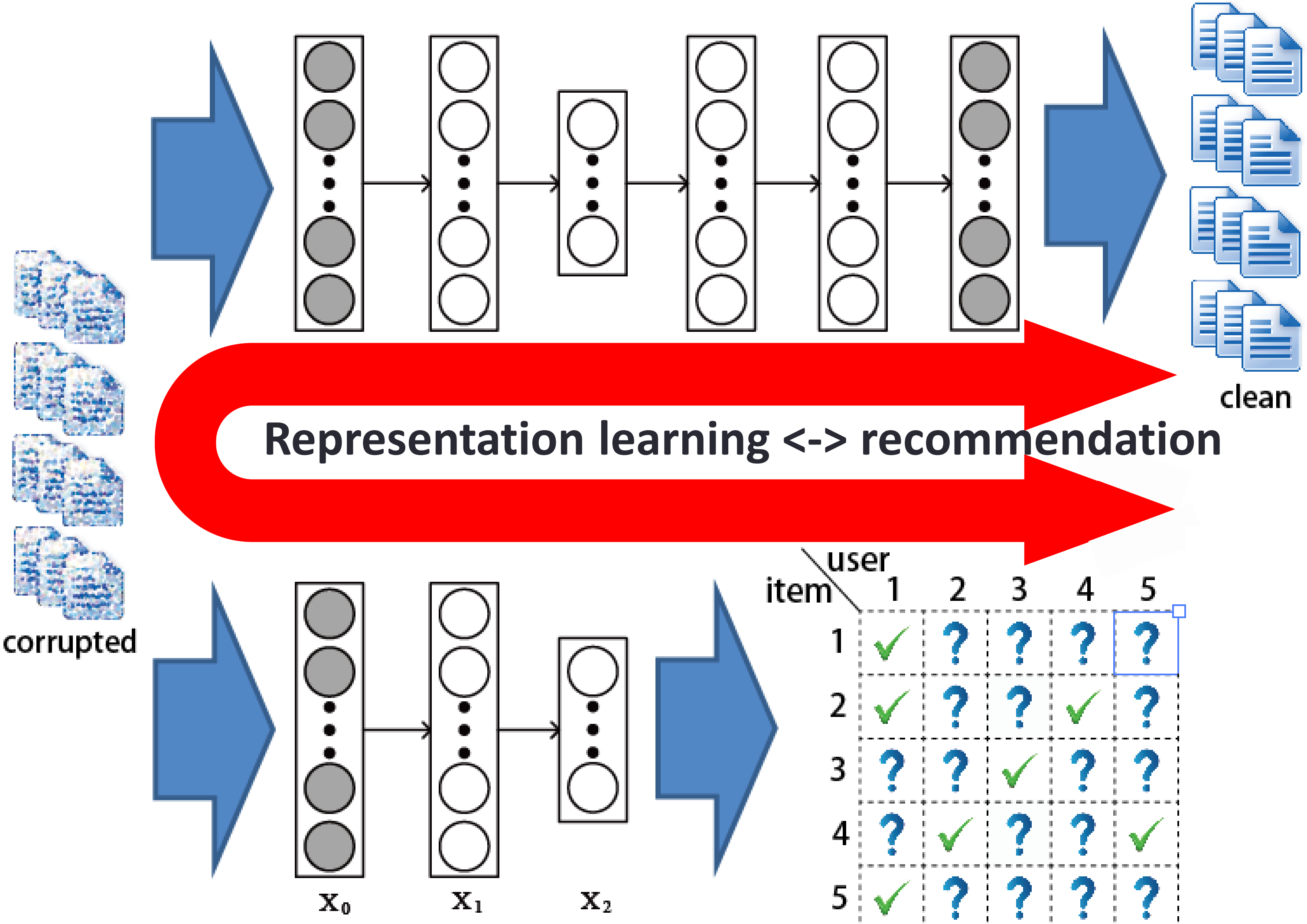
Collaborative Deep Learning



Information flows from content to ratings

item \ user	1	2	3	4	5
1	✓	?	?	?	?
2	✓	?	?	✓	?
3	?	?	✓	?	?
4	?	✓	?	?	✓
5	✓	?	?	?	?

Collaborative Deep Learning



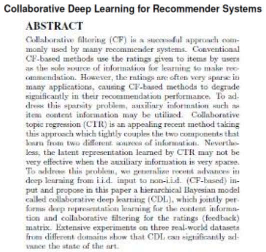
Representation learning <-> recommendation

item \ user	1	2	3	4	5
1	✓	?	?	?	?
2	✓	?	?	✓	?
3	?	?	✓	?	?
4	?	✓	?	?	✓
5	✓	?	?	?	?

Datasets

	citeulike-a	citeulike-t	Netflix
#users	5551	7947	407261
#items	16980	25975	9228
#ratings	204987	134860	15348808

Content information

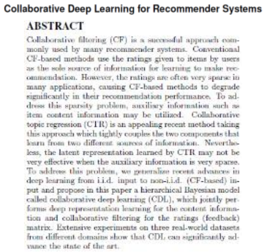


Collaborative Deep Learning for Recommender Systems

ABSTRACT

Collaborative filtering (CF) is a successful approach commonly used by many recommender systems. Conventional CF-based methods use the ratings given to items by users as the sole source of information for learning to make recommendation. However, the ratings are often very sparse in many applications, causing CF-based methods to degrade significantly in their recommendation performance. To address this sparsity problem, auxiliary information such as item content information may be utilized. Collaborative topic regression (CTR) is an appealing recent method taking this approach which tightly couples the two components that learn from two different sources of information. Nevertheless, the latent representation learned by CTR may not be very effective when the auxiliary information is very sparse. To address this problem, we generalize recent advances in deep learning from i.i.d. input to non-i.i.d. (CF-based) input and propose in this paper a hierarchical Bayesian model called collaborative deep learning (CDL), which jointly performs deep representation learning for the content information and collaborative filtering for the ratings (feedback) matrix. Extensive experiments on three real-world datasets from different domains show that CDL can significantly advance the state of the art.

Titles and abstracts

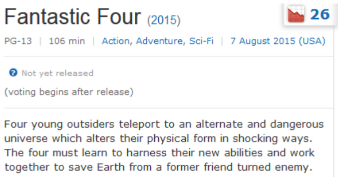


Collaborative Deep Learning for Recommender Systems

ABSTRACT

Collaborative filtering (CF) is a successful approach commonly used by many recommender systems. Conventional CF-based methods use the ratings given to items by users as the sole source of information for learning to make recommendation. However, the ratings are often very sparse in many applications, causing CF-based methods to degrade significantly in their recommendation performance. To address this sparsity problem, auxiliary information such as item content information may be utilized. Collaborative topic regression (CTR) is an appealing recent method taking this approach which tightly couples the two components that learn from two different sources of information. Nevertheless, the latent representation learned by CTR may not be very effective when the auxiliary information is very sparse. To address this problem, we generalize recent advances in deep learning from i.i.d. input to non-i.i.d. (CF-based) input and propose in this paper a hierarchical Bayesian model called collaborative deep learning (CDL), which jointly performs deep representation learning for the content information and collaborative filtering for the ratings (feedback) matrix. Extensive experiments on three real-world datasets from different domains show that CDL can significantly advance the state of the art.

Titles and abstracts



Fantastic Four (2015)

PG-13 | 106 min | Action, Adventure, Sci-Fi | 7 August 2015 (USA)

Not yet released

(voting begins after release)

Four young outsiders teleport to an alternate and dangerous universe which alters their physical form in shocking ways. The four must learn to harness their new abilities and work together to save Earth from a former friend turned enemy.

Movie plots

[Wang et al. KDD 2011]
[**Wang** et al. IJCAI 2013]

Evaluation Metrics

Recall:

$$\text{recall@}M = \frac{\text{number of items that the user likes among the top } M}{\text{total number of items that the user likes}}$$

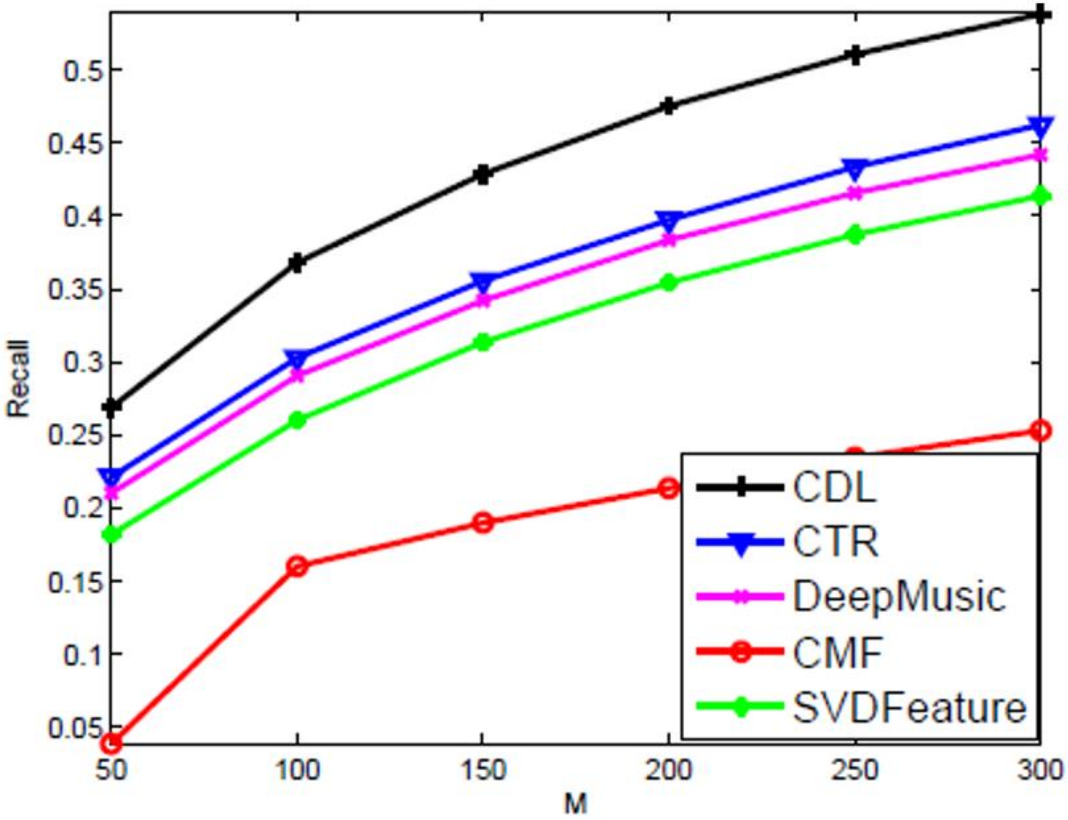
Mean Average Precision (mAP):

$$mAP = \frac{\sum_{q=1}^{Q} AveP(q)}{Q}$$

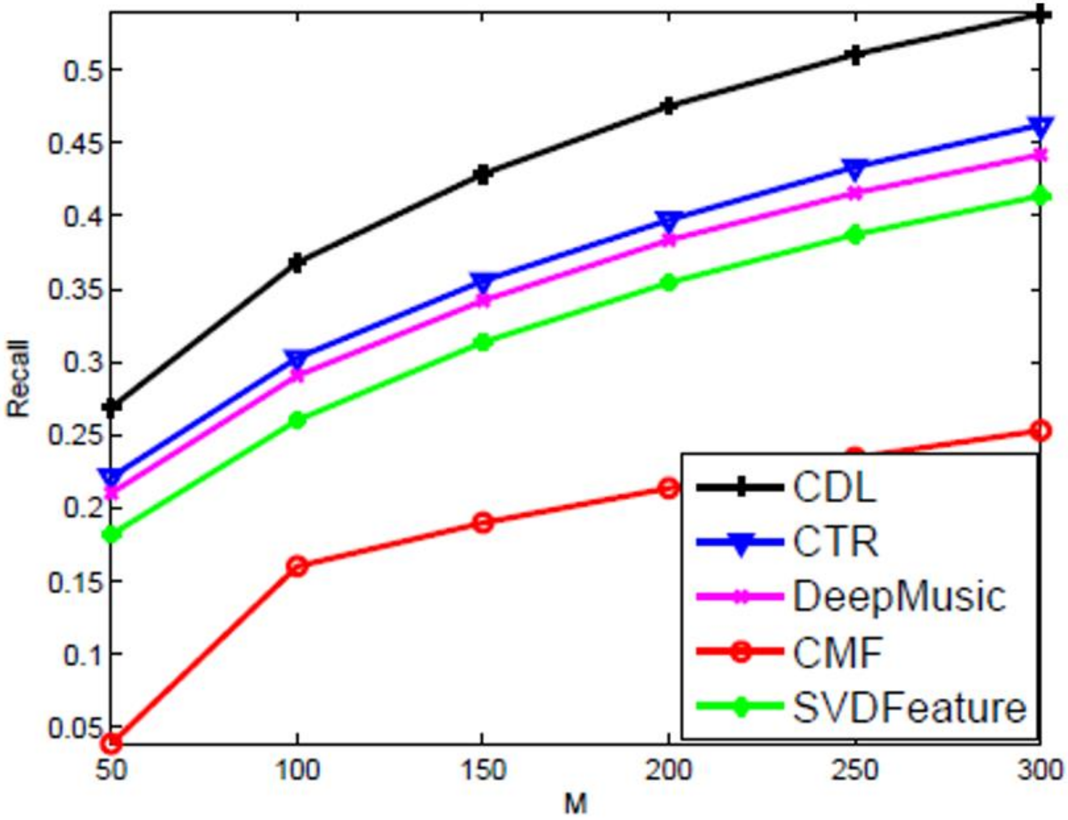
$$AveP = \frac{\sum_{k=1}^{n} (P(k) \times rel(k))}{\text{number of relevant items}}$$

Higher recall and mAP indicate better recommendation performance

Recall@M in *citeulike-t*

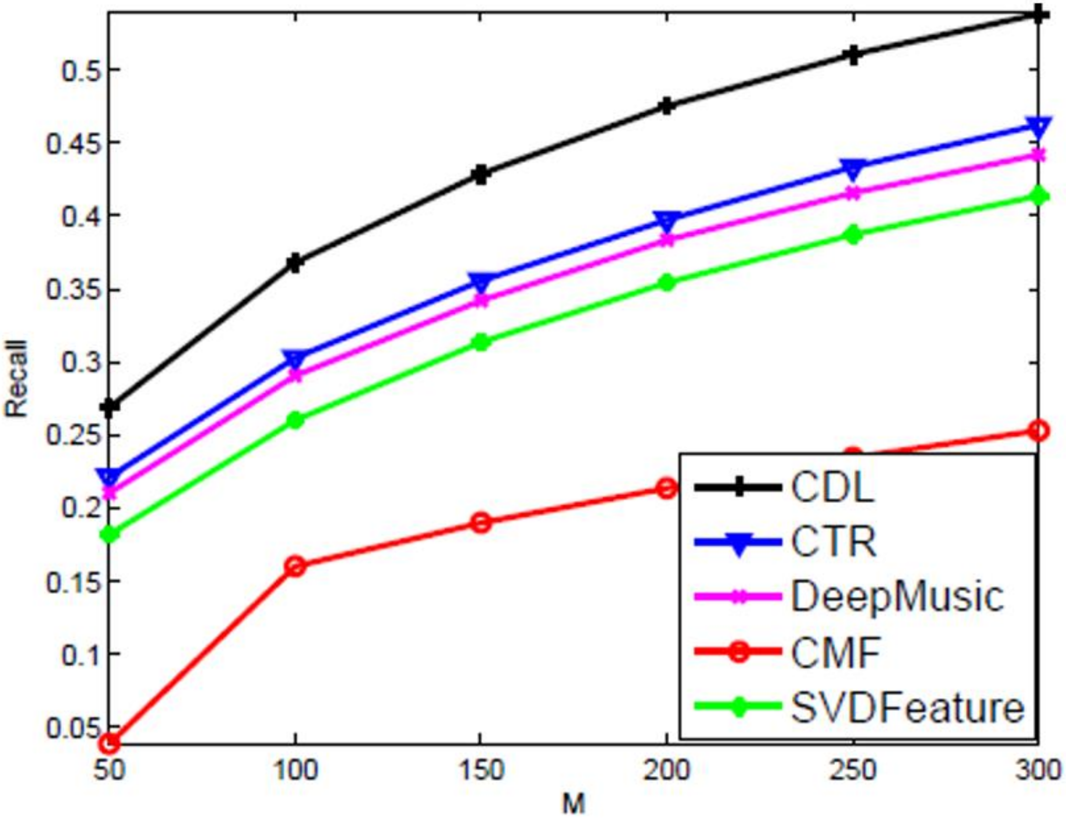


Recall@M in *citeulike-t*



x-axis: number of recommended items M

Recall@M in *citeulike-t*

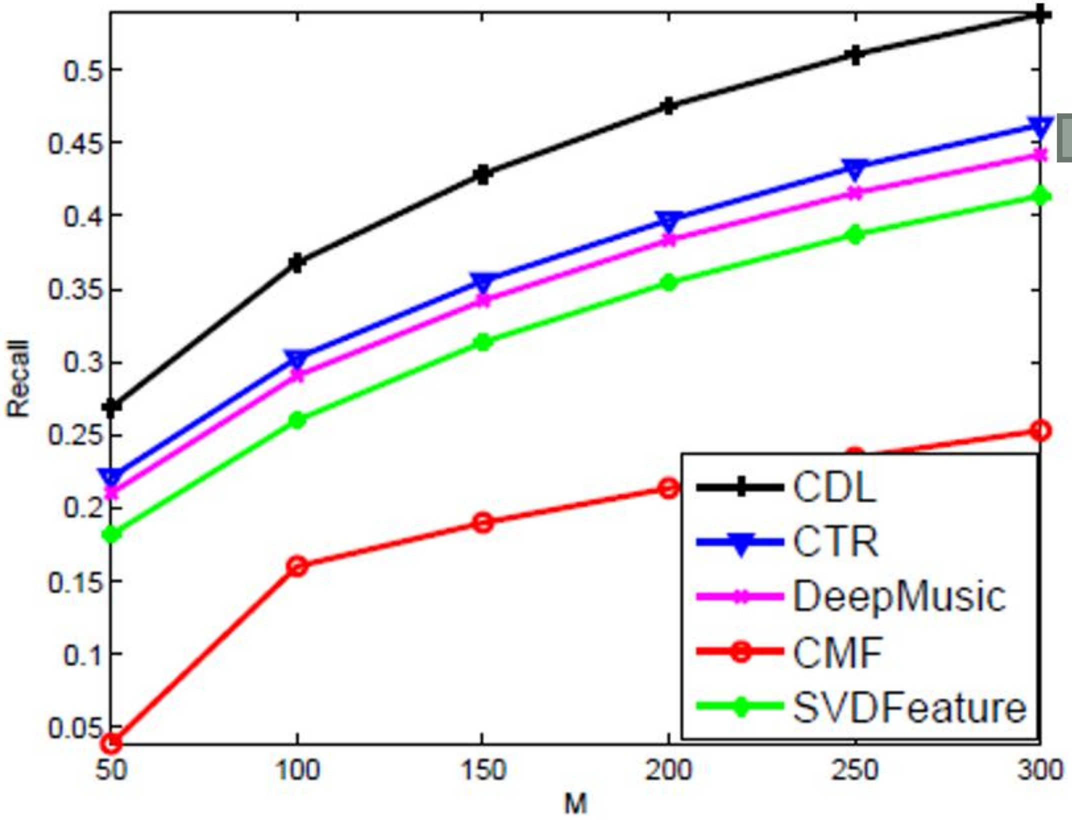


y-axis: recall@M

$$\frac{number\ of\ hits}{total\ correct\ items}$$

x-axis: number of recommended items M

Recall@M in *citeulike-t*



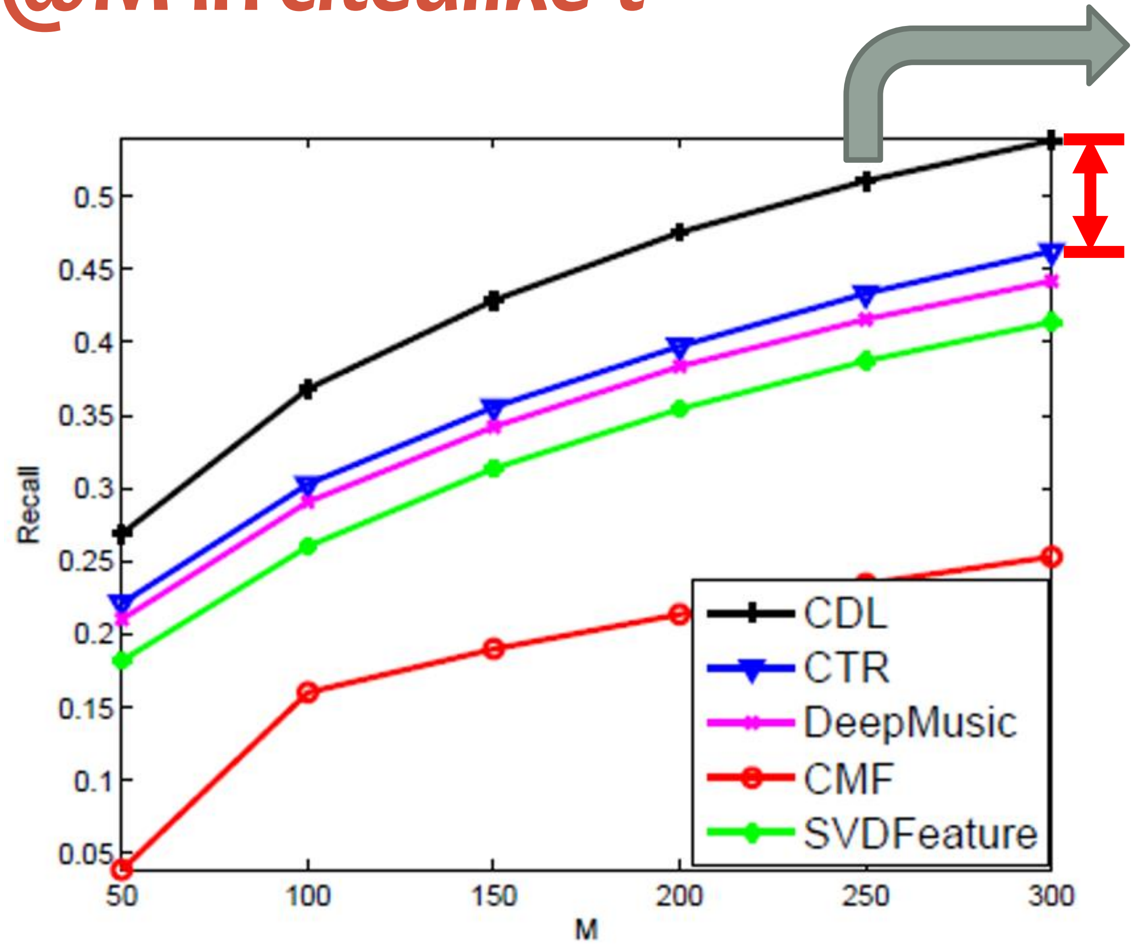
CTR & DeepMusic
best baselines

y-axis:
recall@M

$$\frac{number\ of\ hits}{total\ correct\ items}$$

x-axis: number of recommended items M

Recall@M in *citeulike-t*

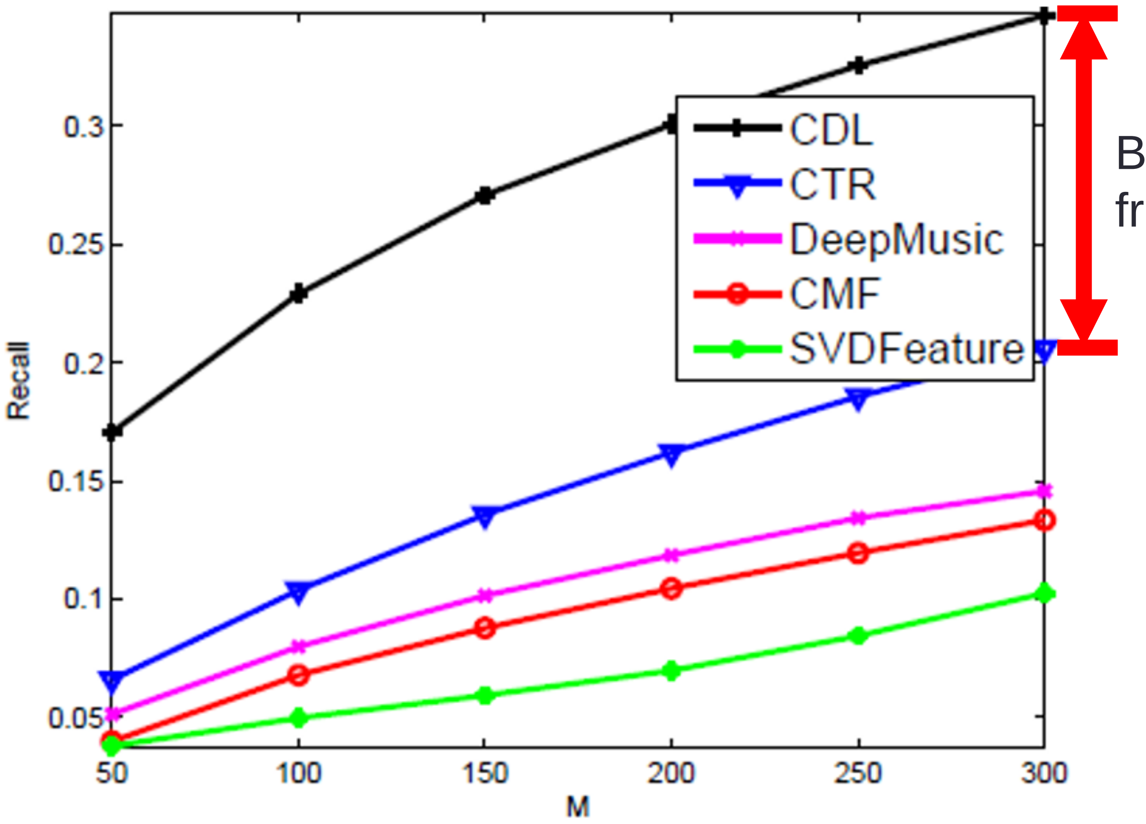


Our method: **CDL**

Boost recall
from **46%** to **54%**

8% absolute improvement

Recall@M in *citeulike-t (sparse ratings)*



Boost recall
from **21%** to **35%**

14% absolute improvement

Sparse ratings

user movie	1	2	3	4	5
1	✓	?	?	?	?
2	✓	?	?	✓	?
3	?	?	✓	?	?
4	?	✓	?	?	✓
5	✓	?	?	?	?

Sparse rating matrix

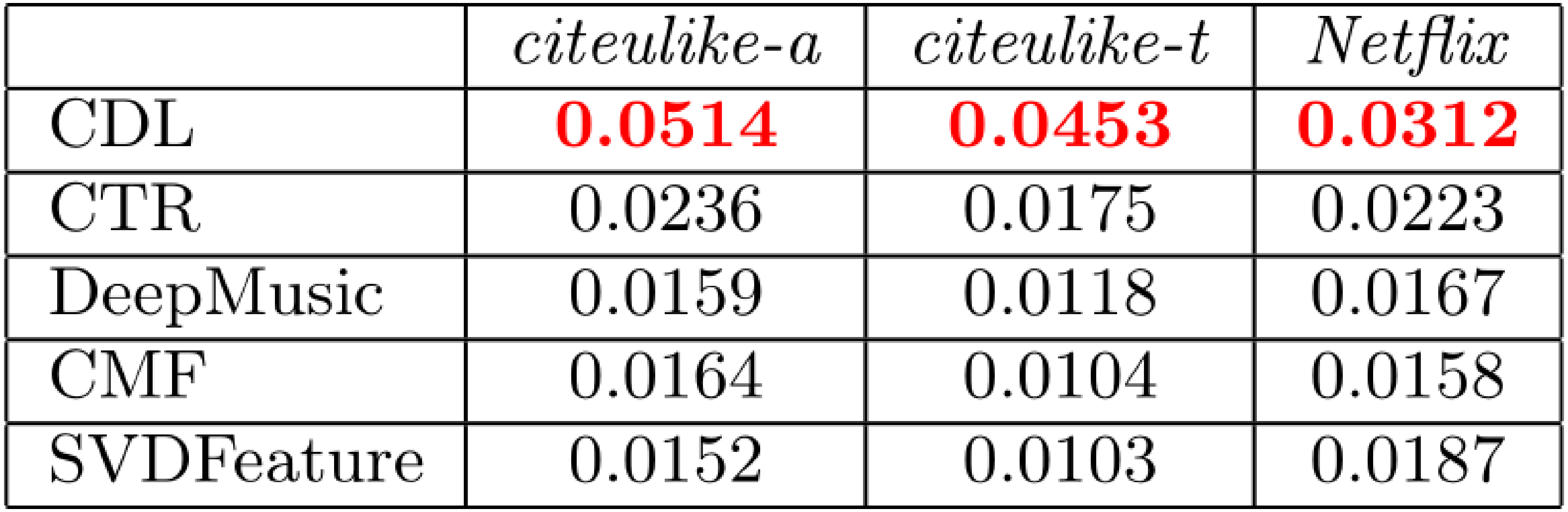
Sparse ratings

movie \ user	1	2	3	4	5
1	✓	?	?	?	?
2	✓	?	?	✓	?
3	?	?	✓	?	?
4	?	✓	?	?	✓
5	✓	?	?	?	?

Content information:
Plots, directors, actors, etc.

Sparse rating matrix

Mean Average Precision (mAP)



	citeulike-a	*citeulike-t*	*Netflix*
CDL	**0.0514**	**0.0453**	**0.0312**
CTR	0.0236	0.0175	0.0223
DeepMusic	0.0159	0.0118	0.0167
CMF	0.0164	0.0104	0.0158
SVDFeature	0.0152	0.0103	0.0187

Exactly the same as Oord et al. 2013, we set the cutoff point at 500 for each user.

A relative performance boost of about 50%

[**Wang** et al. KDD 2015]

Recommender Systems and Revenue

35%

of the revenue comes from recommendations

amazon

[From **McKinsey & Company**]

Recommender Systems and Revenue

$177 Billion × 35% = $62 Billion

(Yearly Sales Revenue)

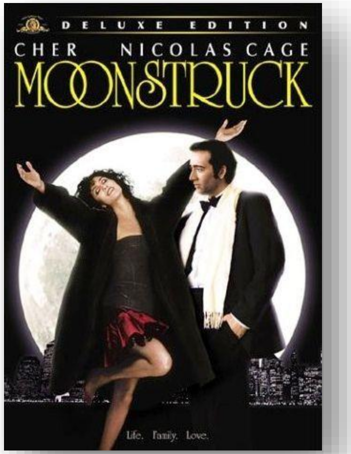
[From **McKinsey & Company**]

Recommender Systems and Revenue

1% → $620 Million

[From **McKinsey & Company**]

Example User



Moonstruck

Romance & Drama Movies

True Romance

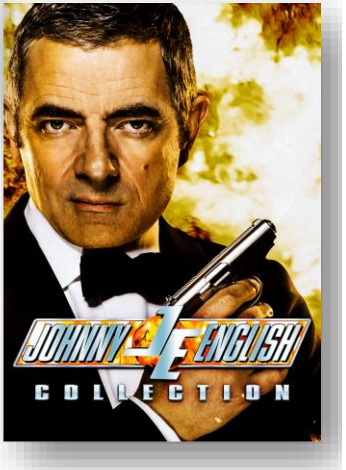
# movies watched	2
Top 10 recommended movies by **CTR** (baseline)	Swordfish
	A Fish Called Wanda
	Terminator 2
	A Clockwork Orange
	Sling Blade
	Bridget Jones's Diary
	Raising Arizona
	A Streetcar Named Desire
	The Untouchables
	The Full Monty

# movies watched	2
Top 10 recommended movies by **CDL** (ours)	Snatch
	The Big Lebowski
	Pulp Fiction
	Kill Bill
	Raising Arizona
	The Big Chill
	Tootsie
	Sense and Sensibility
	Sling Blade
	Swinger

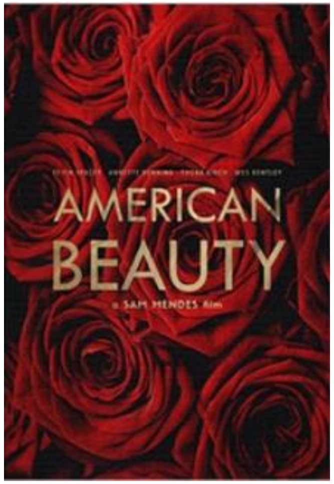
Precision: 20% VS 30%

Example User

Action & Drama Movies



Johnny English



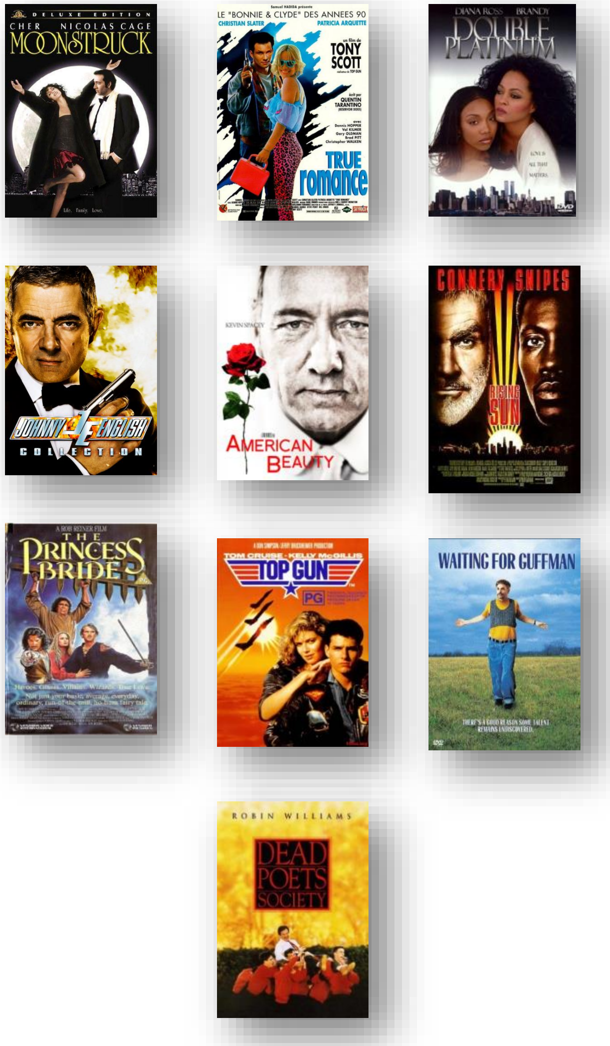
American Beauty

# movies watched	4
Top 10 recommended movies by **CTR** (baseline)	**Pulp Fiction**
	A Clockwork Orange
	Being John Malkovich
	Raising Arizona
	Sling Blade
	Swordfish
	A Fish Called Wanda
	Saving Grace
	The Graduate
	Monster's Ball

# movies watched	4
Top 10 recommended movies by **CDL** (ours)	**Pulp Fiction**
	Snatch
	The Usual Suspect
	Kill Bill
	Momento
	The Big Lebowski
	One Flew Over the Cuckoo's Nest
	As Good as It Gets
	Goodfellas
	The Matrix

Precision: 20% VS 50%

Example User



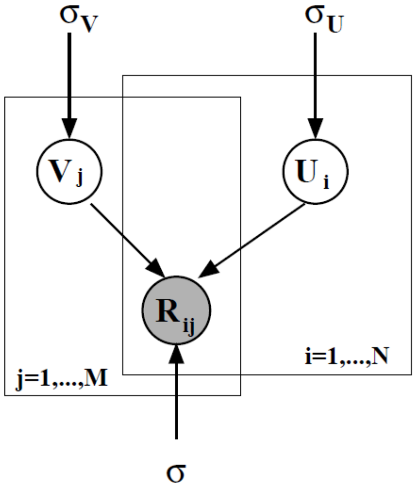
# movies watched	10
Top 10 recommended movies by **CTR** (baseline)	**Best in Snow**
	Chocolat
	Good Will Hunting
	Monty Python and the Holy Grail
	Being John Malkovich
	Raising Arizona
	The Graduate
	Swordfish
	Tootsie
	Saving Private Ryan

# movies watched	10
Top 10 recommended movies by **CDL** (ours)	**Good Will Hunting**
	Best in Show
	The Big Lebowski
	A Few Good Men
	Monty Python and the Holy Grail
	Pulp Fiction
	The Matrix
	Chocolat
	The Usual Suspect
	CaddyShack

Precision: 50% VS 90%

Learning of CDL

Probabilistic Matrix Factorization: Maximum A Posteriori (MAP) Inference (Recap)



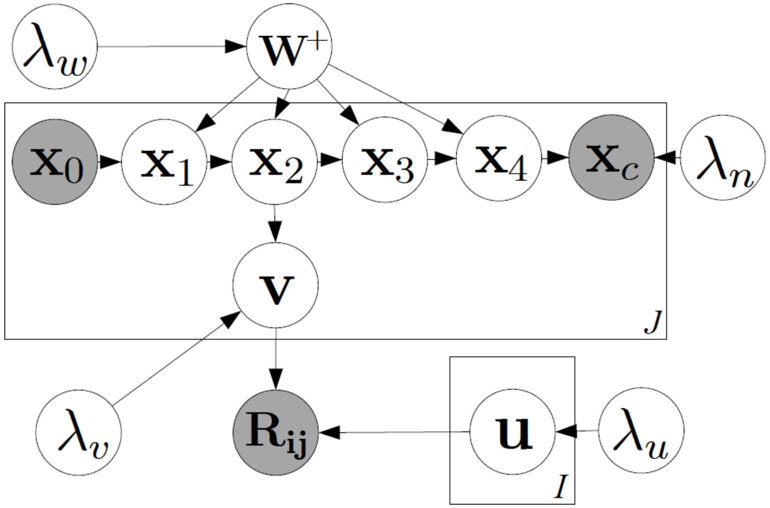
Maximizing the log-posterior over item vectors V_j and user vectors U_i when fixing the hyperparameters (i.e. the observation noise variance σ and prior variances σ_U, σ_V) is equivalent to minimizing the **sum-of-squared-errors** objective function with **quadratic regularization terms**:

$$E = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{M} I_{ij}\left(R_{ij} - U_i^T V_j\right)^2 + \frac{\lambda_U}{2}\sum_{i=1}^{N} \parallel U_i \parallel_{Fro}^2 + \frac{\lambda_V}{2}\sum_{j=1}^{M} \parallel V_j \parallel_{Fro}^2,$$

where $\lambda_U = \sigma^2/\sigma_U$, $\lambda_V = \sigma^2/\sigma_V$, and $||\cdot||_{Fro}$ denotes the Frobenius norm.

Collaborative Deep Learning

Graphical Model



Generative Process

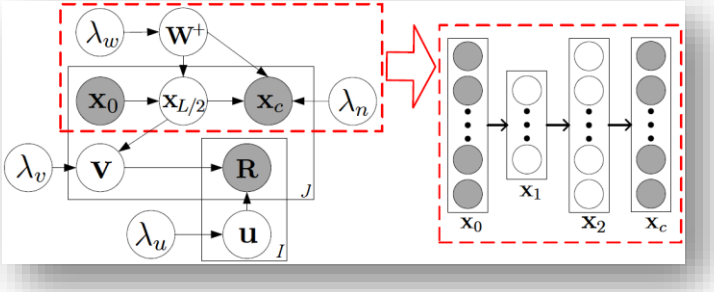
1. For each layer l of the SDAE network,
   (a) For each column n of the weight matrix $\mathbf{W}_l$, draw
   $$\mathbf{W}_{l,*n} \sim \mathcal{N}(\mathbf{0}, \lambda_w^{-1}\mathbf{I}_{K_l}).$$
   (b) Draw the bias vector $\mathbf{b}_l \sim \mathcal{N}(\mathbf{0}, \lambda_w^{-1}\mathbf{I}_{K_l})$.
   (c) For each row j of $\mathbf{X}_l$, draw
   $$\mathbf{X}_{l,j*} \sim \mathcal{N}(\sigma(\mathbf{X}_{l-1,j*}\mathbf{W}_l + \mathbf{b}_l), \lambda_s^{-1}\mathbf{I}_{K_l}).$$
2. For each item j,
   (a) Draw a clean input $\mathbf{X}_{c,j*} \sim \mathcal{N}(\mathbf{X}_{L,j*}, \lambda_n^{-1}\mathbf{I}_J)$.
   (b) Draw a latent item offset vector $\boldsymbol{\epsilon}_j \sim \mathcal{N}(\mathbf{0}, \lambda_v^{-1}\mathbf{I}_K)$ and then set the latent item vector to be:
   $$\mathbf{v}_j = \boldsymbol{\epsilon}_j + \mathbf{X}^T_{\frac{L}{2},j*}.$$
3. Draw a latent user vector for each user i:
   $$\mathbf{u}_i \sim \mathcal{N}(\mathbf{0}, \lambda_u^{-1}\mathbf{I}_K).$$
4. Draw a rating $\mathbf{R}_{ij}$ for each user-item pair (i, j):
   $$\mathbf{R}_{ij} \sim \mathcal{N}(\mathbf{u}_i^T\mathbf{v}_j, \mathbf{C}_{ij}^{-1}).$$

$$E = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{M} I_{ij}(R_{ij} - U_i^T V_j)^2 + \frac{\lambda_U}{2}\sum_{i=1}^{N}\| U_i \|_{Fro}^2 + \frac{\lambda_V}{2}\sum_{j=1}^{M}\| V_j \|_{Fro}^2,$$

Learning

maximizing the posterior probability of U and V is equivalent to maximizing the joint log-likelihood

$$\begin{aligned}\mathscr{L} = & -\frac{\lambda_u}{2}\sum_i \|\mathbf{u}_i\|_2^2 - \frac{\lambda_w}{2}\sum_l (\|\mathbf{W}_l\|_F^2 + \|\mathbf{b}_l\|_2^2) \\ & - \frac{\lambda_v}{2}\sum_j \|\mathbf{v}_j - \mathbf{X}_{\frac{L}{2},j*}^T\|_2^2 - \frac{\lambda_n}{2}\sum_j \|\mathbf{X}_{L,j*} - \mathbf{X}_{c,j*}\|_2^2 \\ & - \frac{\lambda_s}{2}\sum_l\sum_j \|\sigma(\mathbf{X}_{l-1,j*}\mathbf{W}_l + \mathbf{b}_l) - \mathbf{X}_{l,j*}\|_2^2 \\ & - \sum_{i,j} \frac{\mathbf{C}_{ij}}{2}(\mathbf{R}_{ij} - \mathbf{u}_i^T\mathbf{v}_j)^2.\end{aligned}$$



$$E = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{M} I_{ij}(R_{ij} - U_i^T V_j)^2 + \frac{\lambda_U}{2}\sum_{i=1}^{N} \| U_i \|_{Fro}^2 + \frac{\lambda_V}{2}\sum_{j=1}^{M} \| V_j \|_{Fro}^2,$$

Learning

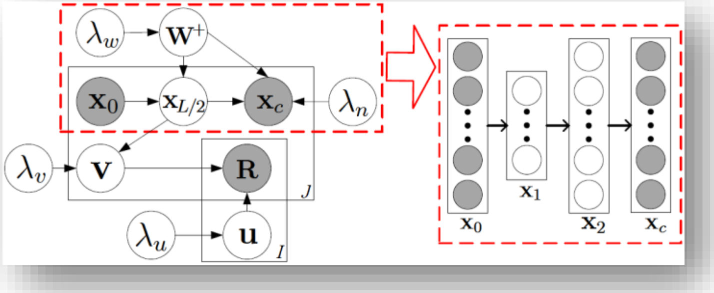
Prior (regularization) for user latent vectors, weights, and biases

$$\mathscr{L} = \boxed{-\frac{\lambda_u}{2}\sum_i \|\mathbf{u}_i\|_2^2 - \frac{\lambda_w}{2}\sum_l (\|\mathbf{W}_l\|_F^2 + \|\mathbf{b}_l\|_2^2)}$$

$$-\frac{\lambda_v}{2}\sum_j \|\mathbf{v}_j - \mathbf{X}^T_{\frac{L}{2},j*}\|_2^2 - \frac{\lambda_n}{2}\sum_j \|\mathbf{X}_{L,j*} - \mathbf{X}_{c,j*}\|_2^2$$

$$-\frac{\lambda_s}{2}\sum_l \sum_j \|\sigma(\mathbf{X}_{l-1,j*}\mathbf{W}_l + \mathbf{b}_l) - \mathbf{X}_{l,j*}\|_2^2$$

$$-\sum_{i,j} \frac{\mathbf{C}_{ij}}{2}(\mathbf{R}_{ij} - \mathbf{u}_i^T \mathbf{v}_j)^2.$$

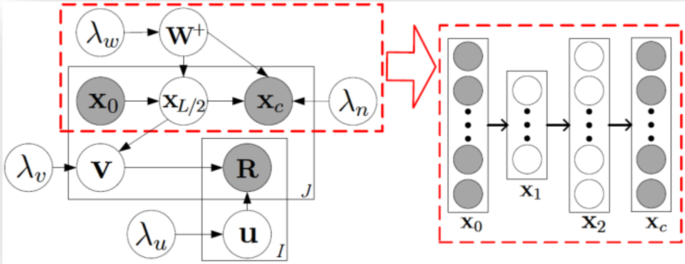


$$E = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{M} I_{ij}\left(R_{ij} - U_i^T V_j\right)^2 + \frac{\lambda_U}{2}\sum_{i=1}^{N} \parallel U_i \parallel_{Fro}^2 + \frac{\lambda_V}{2}\sum_{j=1}^{M} \parallel V_j \parallel_{Fro}^2,$$

Learning

Generating item latent vectors from content representation with Gaussian offset

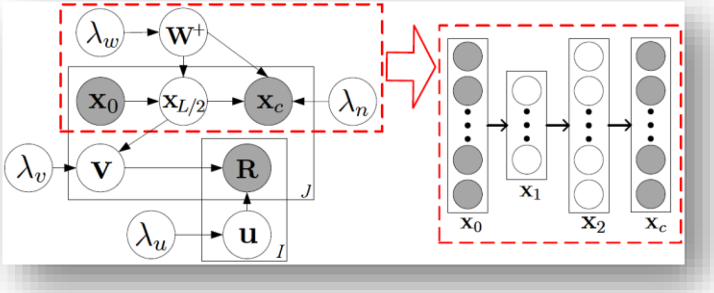
$$\begin{aligned}\mathscr{L} = &-\frac{\lambda_u}{2}\sum_i \|\mathbf{u}_i\|_2^2 - \frac{\lambda_w}{2}\sum_l (\|\mathbf{W}_l\|_F^2 + \|\mathbf{b}_l\|_2^2) \\ &\boxed{-\frac{\lambda_v}{2}\sum_j \|\mathbf{v}_j - \mathbf{X}_{\frac{L}{2},j*}^T\|_2^2} - \frac{\lambda_n}{2}\sum_j \|\mathbf{X}_{L,j*} - \mathbf{X}_{c,j*}\|_2^2 \\ &-\frac{\lambda_s}{2}\sum_l\sum_j \|\sigma(\mathbf{X}_{l-1,j*}\mathbf{W}_l + \mathbf{b}_l) - \mathbf{X}_{l,j*}\|_2^2 \\ &-\sum_{i,j}\frac{\mathbf{C}_{ij}}{2}(\mathbf{R}_{ij} - \mathbf{u}_i^T\mathbf{v}_j)^2.\end{aligned}$$



Learning

‘Generating’ clean input from the output of probabilistic SDAE with Gaussian offset

$$
\begin{aligned}
\mathscr{L} = &-\frac{\lambda_u}{2}\sum_i \|\mathbf{u}_i\|_2^2 - \frac{\lambda_w}{2}\sum_l (\|\mathbf{W}_l\|_F^2 + \|\mathbf{b}_l\|_2^2) \\
&- \frac{\lambda_v}{2}\sum_j \|\mathbf{v}_j - \mathbf{X}^T_{\frac{L}{2},j*}\|_2^2 - \boxed{\frac{\lambda_n}{2}\sum_j \|\mathbf{X}_{L,j*} - \mathbf{X}_{c,j*}\|_2^2} \\
&- \frac{\lambda_s}{2}\sum_l\sum_j \|\sigma(\mathbf{X}_{l-1,j*}\mathbf{W}_l + \mathbf{b}_l) - \mathbf{X}_{l,j*}\|_2^2 \\
&- \sum_{i,j}\frac{\mathbf{C}_{ij}}{2}(\mathbf{R}_{ij} - \mathbf{u}_i^T\mathbf{v}_j)^2.
\end{aligned}
$$



Learning

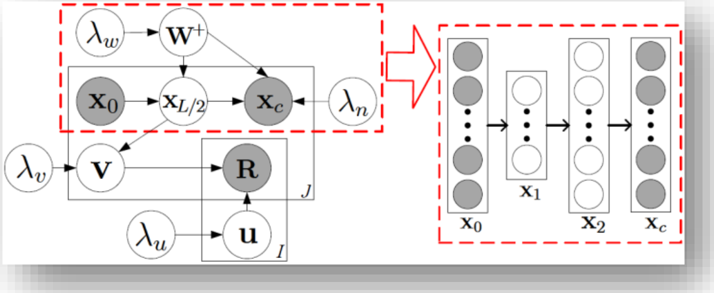
Generating the input of Layer l from the output of Layer l-1 with Gaussian offset

$$\mathscr{L} = -\frac{\lambda_u}{2}\sum_i \|\mathbf{u}_i\|_2^2 - \frac{\lambda_w}{2}\sum_l (\|\mathbf{W}_l\|_F^2 + \|\mathbf{b}_l\|_2^2)$$

$$-\frac{\lambda_v}{2}\sum_j \|\mathbf{v}_j - \mathbf{X}^T_{\frac{L}{2},j*}\|_2^2 - \frac{\lambda_n}{2}\sum_j \|\mathbf{X}_{L,j*} - \mathbf{X}_{c,j*}\|_2^2$$

$$\boxed{-\frac{\lambda_s}{2}\sum_l\sum_j \|\sigma(\mathbf{X}_{l-1,j*}\mathbf{W}_l + \mathbf{b}_l) - \mathbf{X}_{l,j*}\|_2^2}$$

$$-\sum_{i,j}\frac{\mathbf{C}_{ij}}{2}(\mathbf{R}_{ij} - \mathbf{u}_i^T\mathbf{v}_j)^2.$$

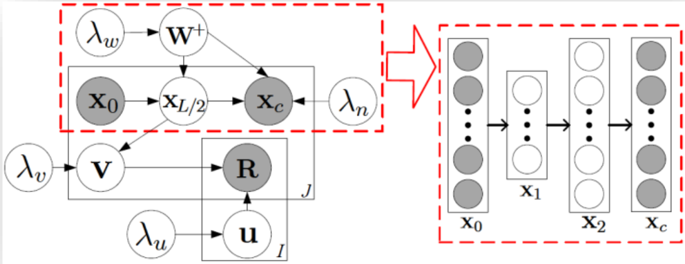


Learning

$$E = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{M} I_{ij}\left(R_{ij} - U_i^T V_j\right)^2 + \frac{\lambda_U}{2}\sum_{i=1}^{N} \| U_i \|_{Fro}^2 + \frac{\lambda_V}{2}\sum_{j=1}^{M} \| V_j \|_{Fro}^2,$$

measures the error of predicted ratings

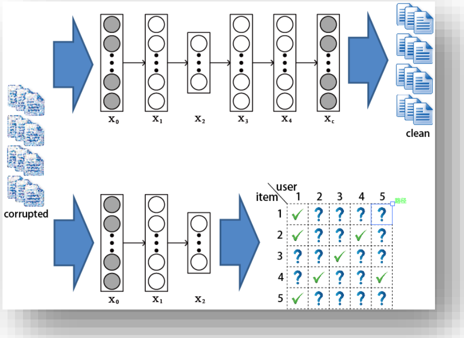
$$\begin{aligned}\mathscr{L} = & -\frac{\lambda_u}{2}\sum_i \|\mathbf{u}_i\|_2^2 - \frac{\lambda_w}{2}\sum_l (\|\mathbf{W}_l\|_F^2 + \|\mathbf{b}_l\|_2^2) \\ & -\frac{\lambda_v}{2}\sum_j \|\mathbf{v}_j - \mathbf{X}_{\frac{L}{2},j*}^T\|_2^2 - \frac{\lambda_n}{2}\sum_j \|\mathbf{X}_{L,j*} - \mathbf{X}_{c,j*}\|_2^2 \\ & -\frac{\lambda_s}{2}\sum_l \sum_j \|\sigma(\mathbf{X}_{l-1,j*}\mathbf{W}_l + \mathbf{b}_l) - \mathbf{X}_{l,j*}\|_2^2 \\ & \boxed{-\sum_{i,j} \frac{\mathbf{C}_{ij}}{2}(\mathbf{R}_{ij} - \mathbf{u}_i^T \mathbf{v}_j)^2}.\end{aligned}$$



Learning

If λ_s goes to infinity, the likelihood simplifies to

$$\begin{aligned}\mathscr{L} = & -\frac{\lambda_u}{2}\sum_i \|\mathbf{u}_i\|_2^2 - \frac{\lambda_w}{2}\sum_l (\|\mathbf{W}_l\|_F^2 + \|\mathbf{b}_l\|_2^2) \\ & -\frac{\lambda_v}{2}\sum_j \|\mathbf{v}_j - f_e(\mathbf{X}_{0,j*}, \mathbf{W}^+)^T\|_2^2 \\ & -\frac{\lambda_n}{2}\sum_j \|f_r(\mathbf{X}_{0,j*}, \mathbf{W}^+) - \mathbf{X}_{c,j*}\|_2^2 \\ & -\sum_{i,j} \frac{\mathbf{C}_{ij}}{2}(\mathbf{R}_{ij} - \mathbf{u}_i^T\mathbf{v}_j)^2,\end{aligned}$$



item \ user	1	2	3	4	5
1	✓	?	?	?	?
2	✓	?	?	✓	?
3	?	?	✓	?	?
4	?	✓	?	?	✓
5	✓	?	?	?	?

Update Rules

For U and V, use block coordinate descent:

$$\mathbf{u}_i \leftarrow (\mathbf{V}\mathbf{C}_i\mathbf{V}^T + \lambda_u \mathbf{I}_K)^{-1}\mathbf{V}\mathbf{C}_i\mathbf{R}_i$$

$$\mathbf{v}_j \leftarrow (\mathbf{U}\mathbf{C}_i\mathbf{U}^T + \lambda_v \mathbf{I}_K)^{-1}(\mathbf{U}\mathbf{C}_j\mathbf{R}_j + \boxed{\lambda_v f_e(\mathbf{X}_{0,j*}, \mathbf{W}^+)^T})$$

For W and b, use a modified version of backpropagation:

$$\nabla_{\mathbf{W}_l}\mathscr{L} = -\lambda_w \mathbf{W}_l$$

$$\boxed{- \lambda_v \sum_j \nabla_{\mathbf{W}_l} f_e(\mathbf{X}_{0,j*}, \mathbf{W}^+)^T (f_e(\mathbf{X}_{0,j*}, \mathbf{W}^+)^T - \mathbf{v}_j)}$$

$$- \lambda_n \sum_j \nabla_{\mathbf{W}_l} f_r(\mathbf{X}_{0,j*}, \mathbf{W}^+)(f_r(\mathbf{X}_{0,j*}, \mathbf{W}^+) - \mathbf{X}_{c,j*})$$

$$\nabla_{\mathbf{b}_l}\mathscr{L} = -\lambda_w \mathbf{b}_l$$

$$\boxed{- \lambda_v \sum_j \nabla_{\mathbf{b}_l} f_e(\mathbf{X}_{0,j*}, \mathbf{W}^+)^T (f_e(\mathbf{X}_{0,j*}, \mathbf{W}^+)^T - \mathbf{v}_j)}$$

$$- \lambda_n \sum_j \nabla_{\mathbf{b}_l} f_r(\mathbf{X}_{0,j*}, \mathbf{W}^+)(f_r(\mathbf{X}_{0,j*}, \mathbf{W}^+) - \mathbf{X}_{c,j*})$$

Brief Introduction for Extensions of CDL/CRAE

Collaborative Deep Ranking

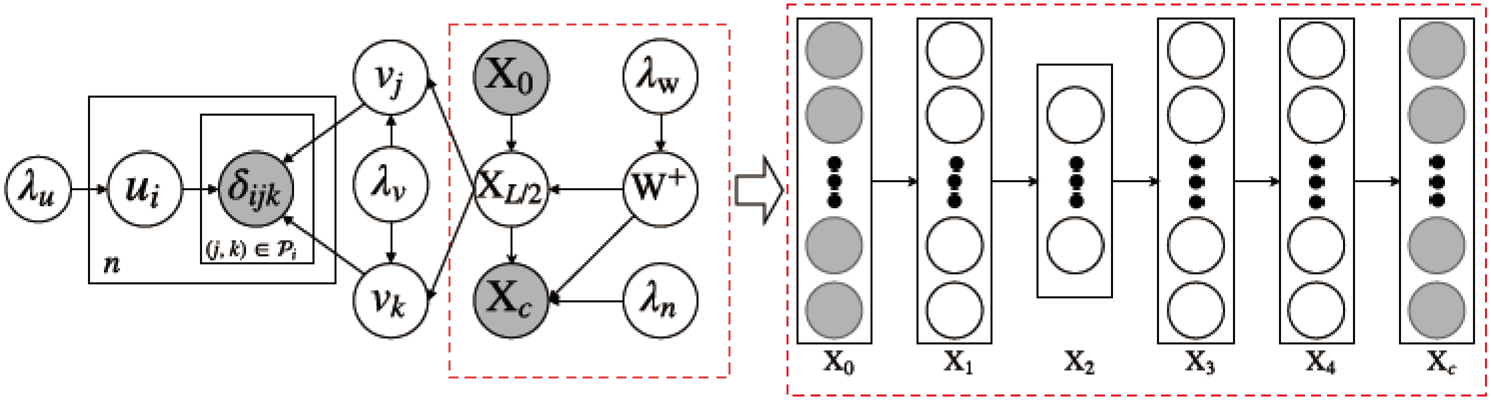
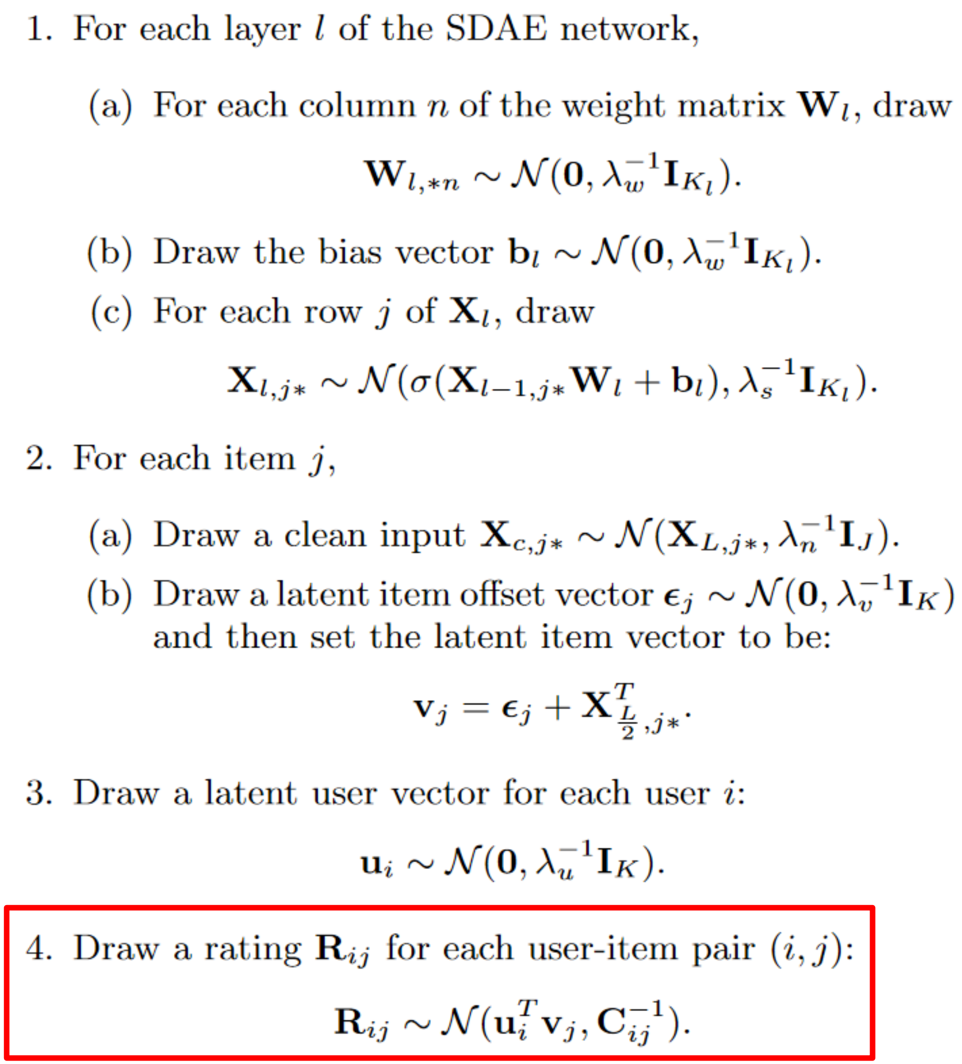


Fig. 1. The graphic model of CDR. SDAE with $L = 4$ is presented inside the dashed rectangle. Note that W^+ denotes the set of weight matrices and bias vectors of all layers.

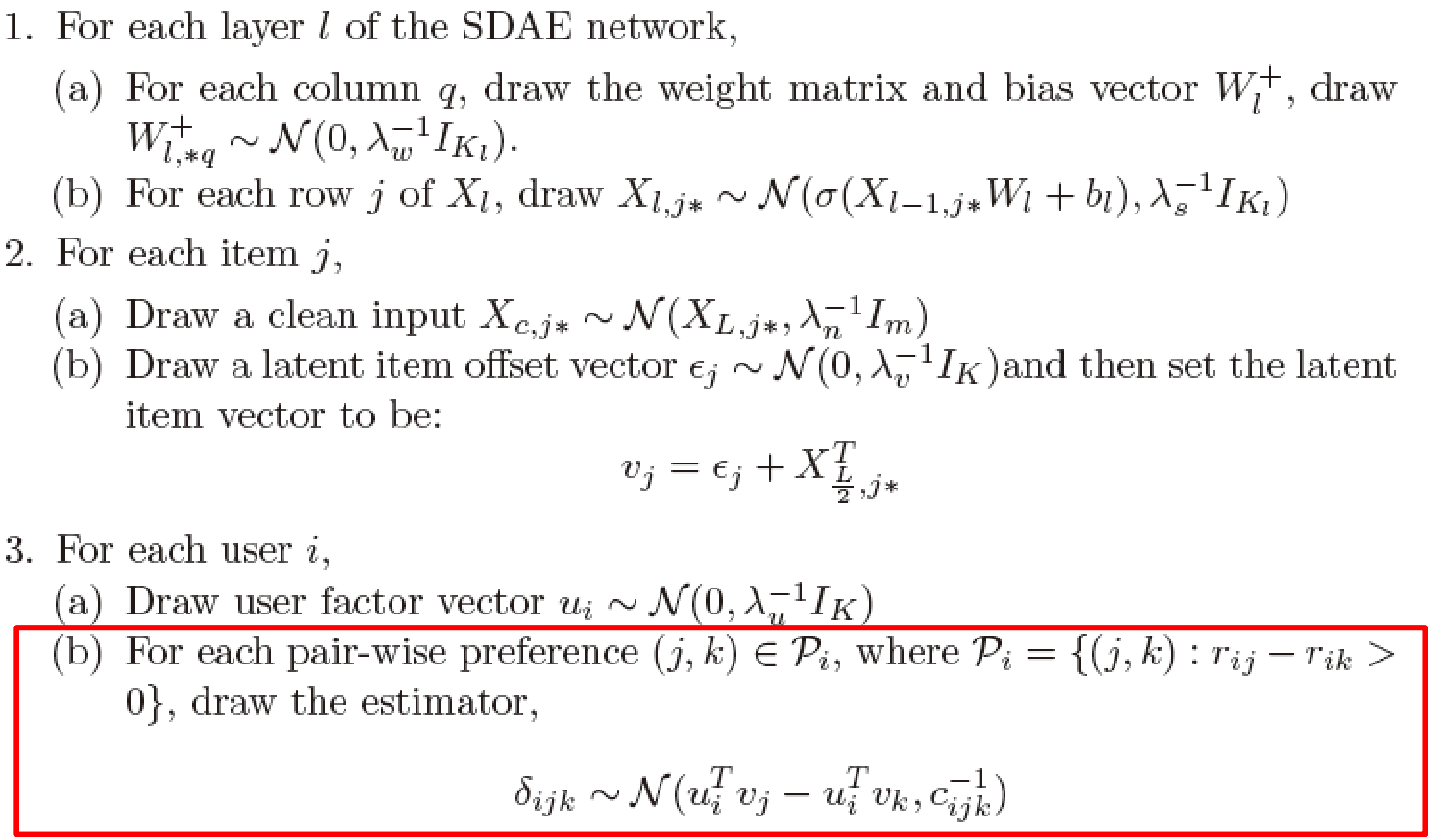
[Ying et al., PAKDD 2016]

Generative Process: Collaborative Deep Learning (Recap)



1. For each layer l of the SDAE network,
   (a) For each column n of the weight matrix $\mathbf{W}_l$, draw
   $$\mathbf{W}_{l,*n} \sim \mathcal{N}(\mathbf{0}, \lambda_w^{-1}\mathbf{I}_{K_l}).$$
   (b) Draw the bias vector $\mathbf{b}_l \sim \mathcal{N}(\mathbf{0}, \lambda_w^{-1}\mathbf{I}_{K_l})$.
   (c) For each row j of $\mathbf{X}_l$, draw
   $$\mathbf{X}_{l,j*} \sim \mathcal{N}(\sigma(\mathbf{X}_{l-1,j*}\mathbf{W}_l + \mathbf{b}_l), \lambda_s^{-1}\mathbf{I}_{K_l}).$$
2. For each item j,
   (a) Draw a clean input $\mathbf{X}_{c,j*} \sim \mathcal{N}(\mathbf{X}_{L,j*}, \lambda_n^{-1}\mathbf{I}_J)$.
   (b) Draw a latent item offset vector $\boldsymbol{\epsilon}_j \sim \mathcal{N}(\mathbf{0}, \lambda_v^{-1}\mathbf{I}_K)$ and then set the latent item vector to be:
   $$\mathbf{v}_j = \boldsymbol{\epsilon}_j + \mathbf{X}^T_{\frac{L}{2},j*}.$$
3. Draw a latent user vector for each user i:
   $$\mathbf{u}_i \sim \mathcal{N}(\mathbf{0}, \lambda_u^{-1}\mathbf{I}_K).$$
4. Draw a rating $\mathbf{R}_{ij}$ for each user-item pair (i, j):
   $$\mathbf{R}_{ij} \sim \mathcal{N}(\mathbf{u}_i^T\mathbf{v}_j, \mathbf{C}_{ij}^{-1}).$$

Generative Process: Collaborative Deep Ranking



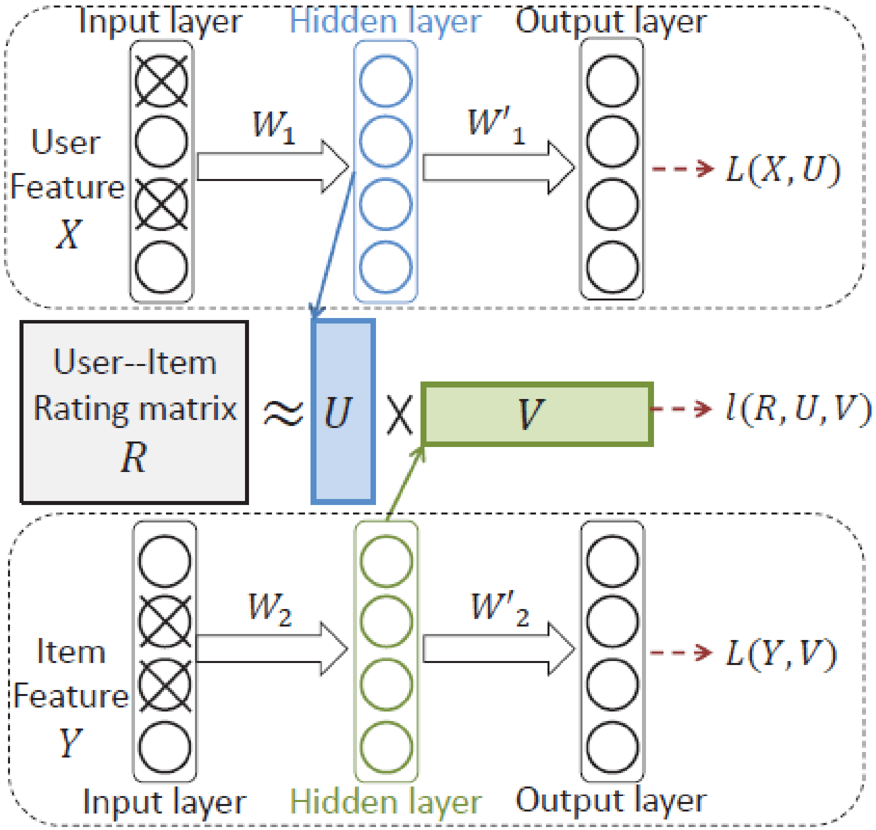
1. For each layer l of the SDAE network,
   - (a) For each column q, draw the weight matrix and bias vector W_l^+, draw $W_{l,*q}^+ \sim \mathcal{N}(0, \lambda_w^{-1} I_{K_l})$.
   - (b) For each row j of X_l, draw $X_{l,j*} \sim \mathcal{N}(\sigma(X_{l-1,j*} W_l + b_l), \lambda_s^{-1} I_{K_l})$
2. For each item j,
   - (a) Draw a clean input $X_{c,j*} \sim \mathcal{N}(X_{L,j*}, \lambda_n^{-1} I_m)$
   - (b) Draw a latent item offset vector $\epsilon_j \sim \mathcal{N}(0, \lambda_v^{-1} I_K)$and then set the latent item vector to be:

$$v_j = \epsilon_j + X^T_{\frac{L}{2},j*}$$

3. For each user i,
   - (a) Draw user factor vector $u_i \sim \mathcal{N}(0, \lambda_u^{-1} I_K)$
   - (b) For each pair-wise preference $(j,k) \in \mathcal{P}_i$, where $\mathcal{P}_i = \{(j,k) : r_{ij} - r_{ik} > 0\}$, draw the estimator,

$$\delta_{ijk} \sim \mathcal{N}(u_i^T v_j - u_i^T v_k, c_{ijk}^{-1})$$

Symmetric CDL



Both item content and user attributes

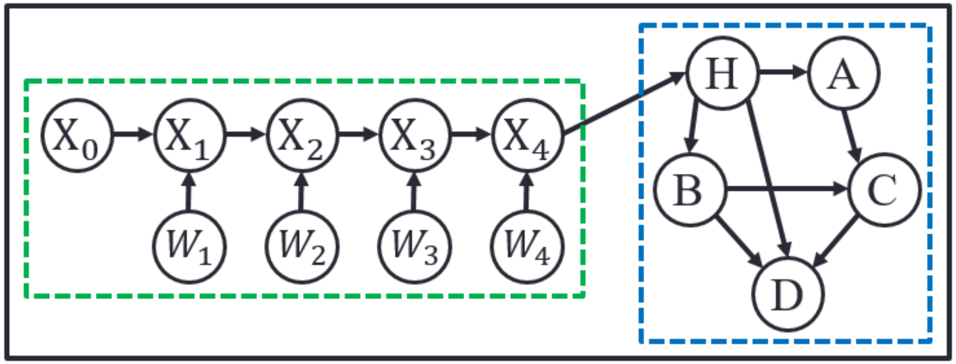
User attributes: age, gender, occupation, country, city, geolacation, domain, etc

[Li et al., CIKM 2015]

Other Extensions of CDL

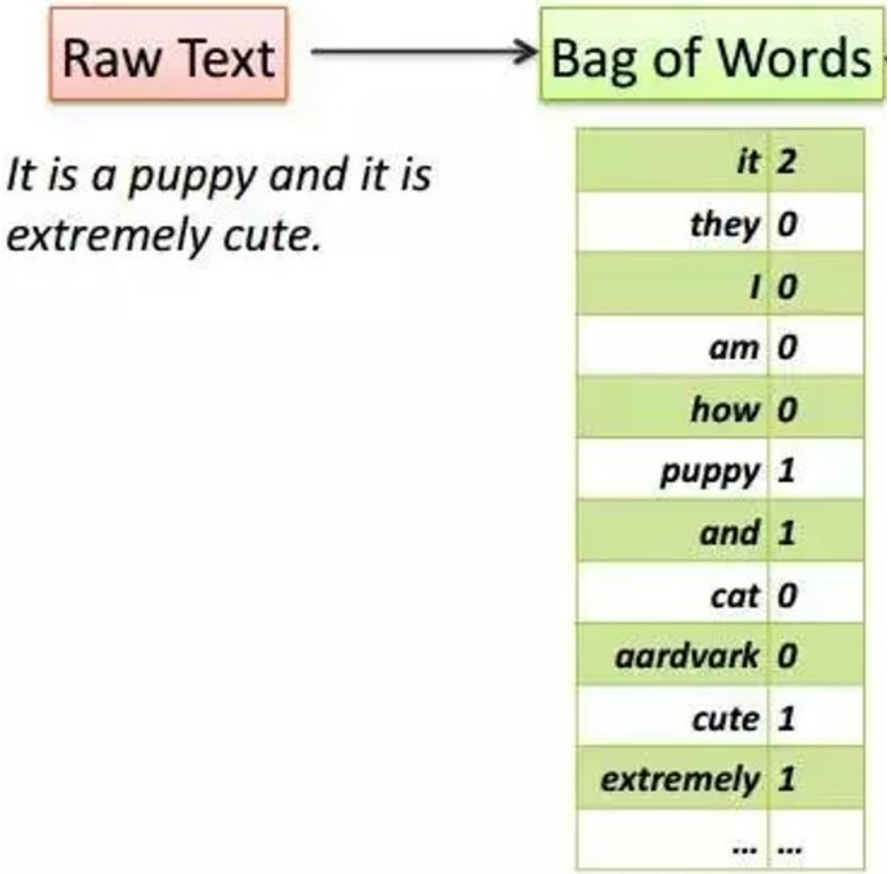
- **Word2vec, tf-idf**
- **Sampling-based, variational inference**
- **Tagging information, networks**

Summary of Collaborative Deep Learning



- A new probabilistic formulation for deep learning models (Challenge 1)
- First hierarchical Bayesian models for deep hybrid recommender systems (Challenge 2)
- Significant performance improvement over the state of the art

Beyond Bag-of-Words



Raw Text → Bag of Words

It is a puppy and it is extremely cute.

it	*2*
they	*0*
I	*0*
am	*0*
how	*0*
puppy	*1*
and	*1*
cat	*0*
aardvark	*0*
cute	*1*
extremely	*1*
...	...

High dimensional sparse vector

Bag-of-Words:

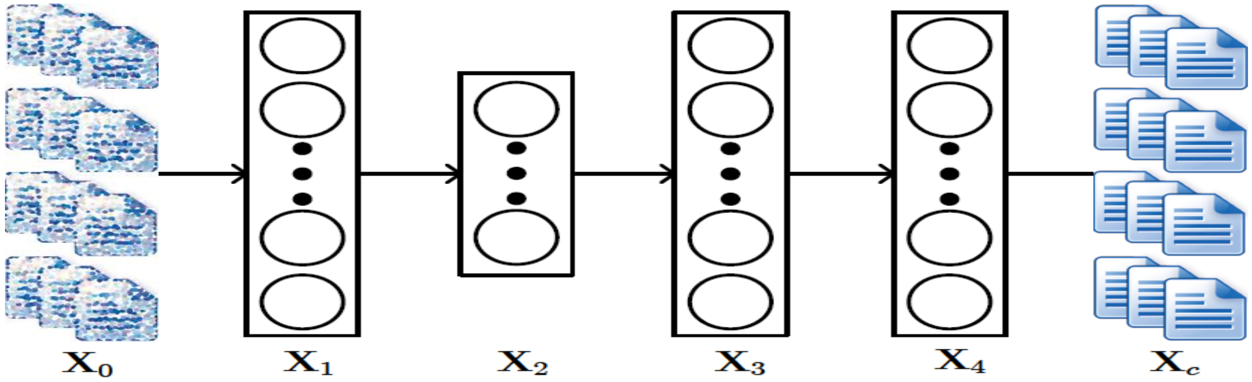
- Ignore word order
- No local context

Instead of Bag-of-Words

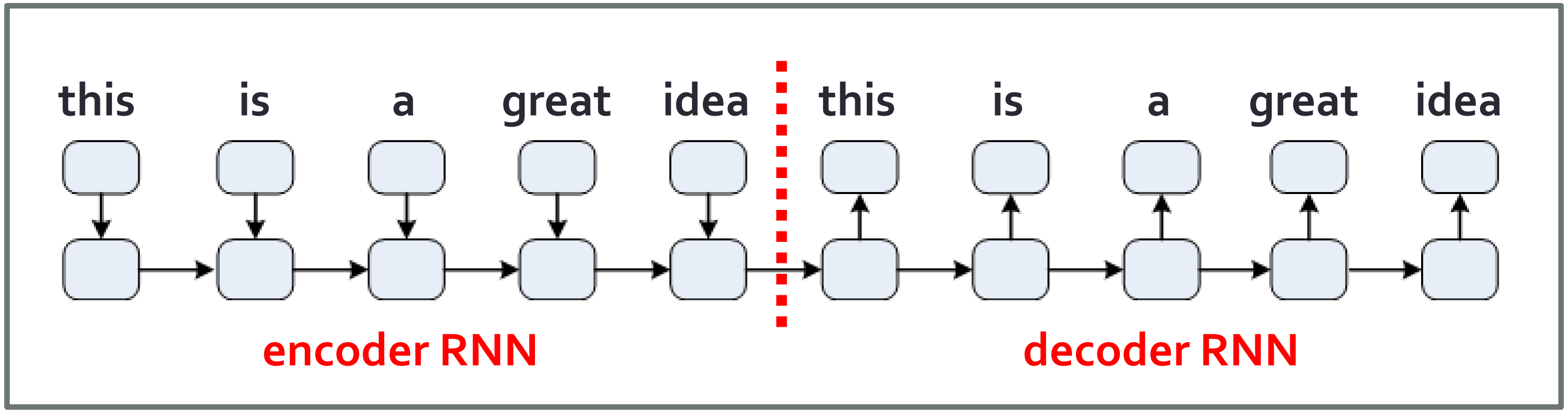
Want representation
- Aware of **sequential** relation of words
- Robust to **missing** words

Document as a Sequence

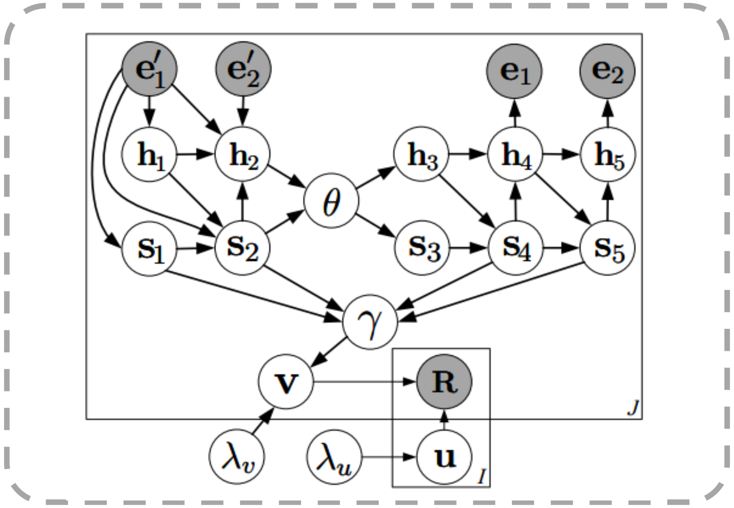
Feedforward autoencoder



Recurrent autoencoder



Challenge 1: Encoder Learns Wrong Transition



"Collaborative recurrent autoencoder: recommend while learning to fill in the blanks" [**Wang** et al., NIPS 2016a]

Challenges:

- RNN encoder may learn wrong transition between words

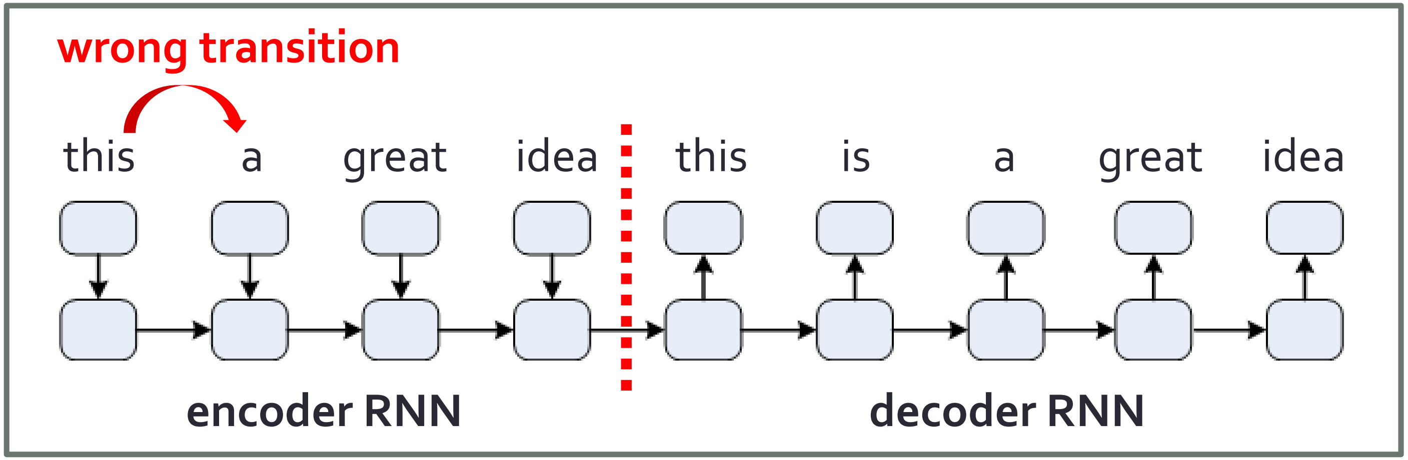
Challenge 1: Encoder Learns Wrong Transition

Sentence: This ~~is~~ a great idea.

Challenge 1: Encoder Learns Wrong Transition

Sentence: This ~~is~~ a great idea. -> This is a great idea.

Direct Denoising:

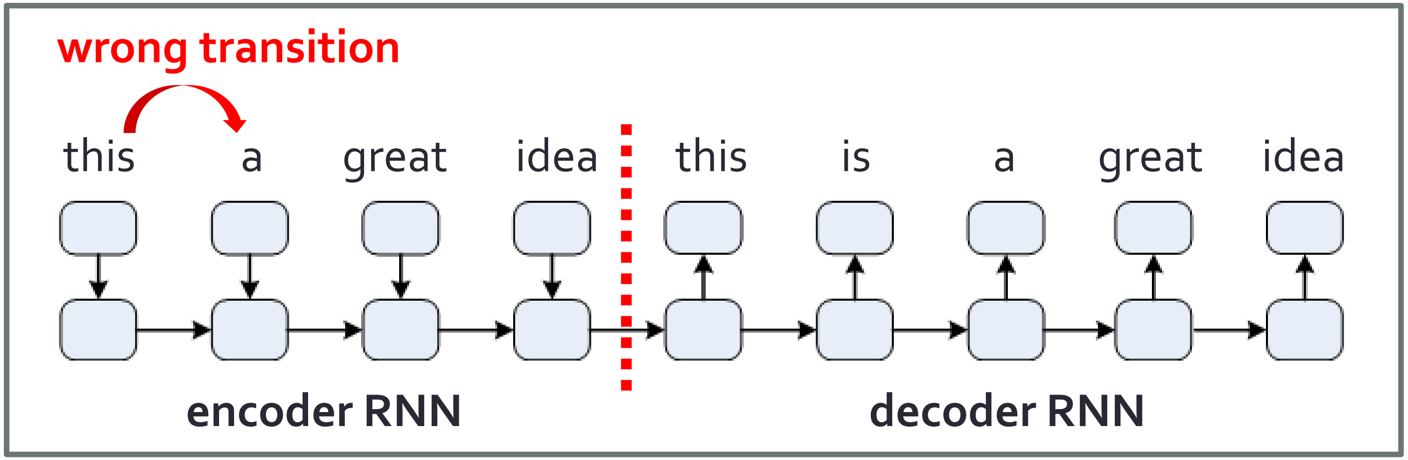


RNN encoder learns wrong transition between ‘**this**’ and ‘**a**’

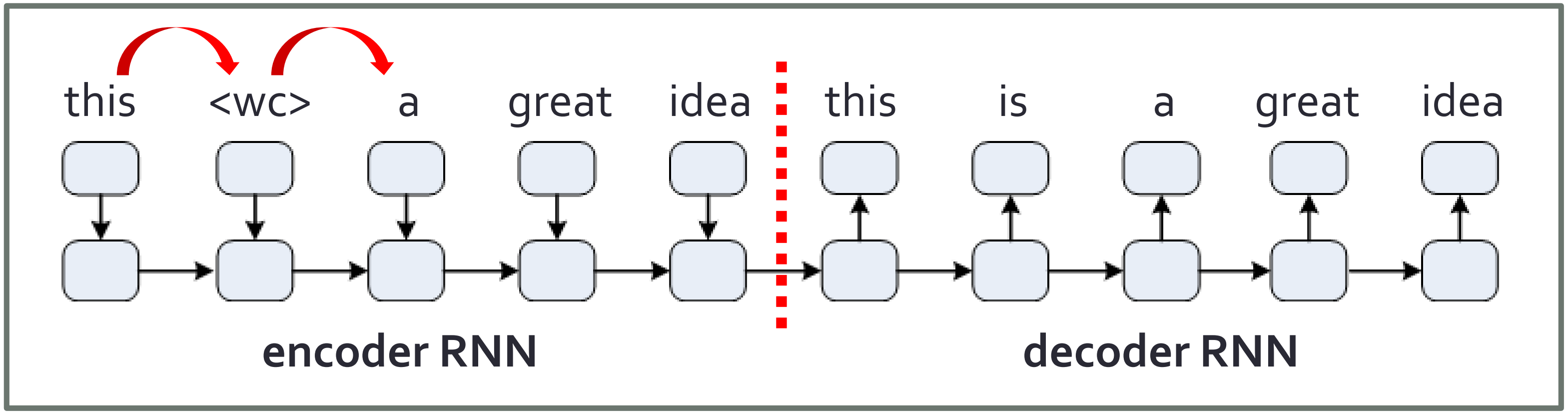
Wildcard Denoising: Avoiding Wrong Transition

Sentence: This ~~is~~ a great idea. -> This is a great idea.

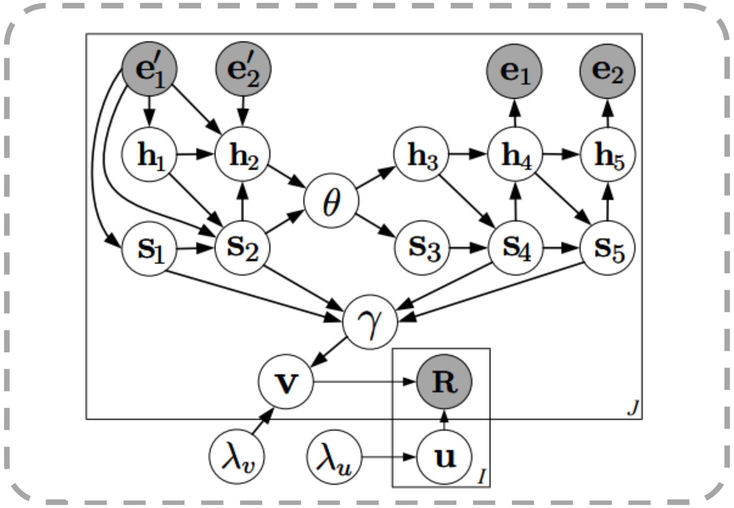
Direct Denoising:



Wildcard Denoising:



Challenge 2: Variable-Length Vector for Pooling



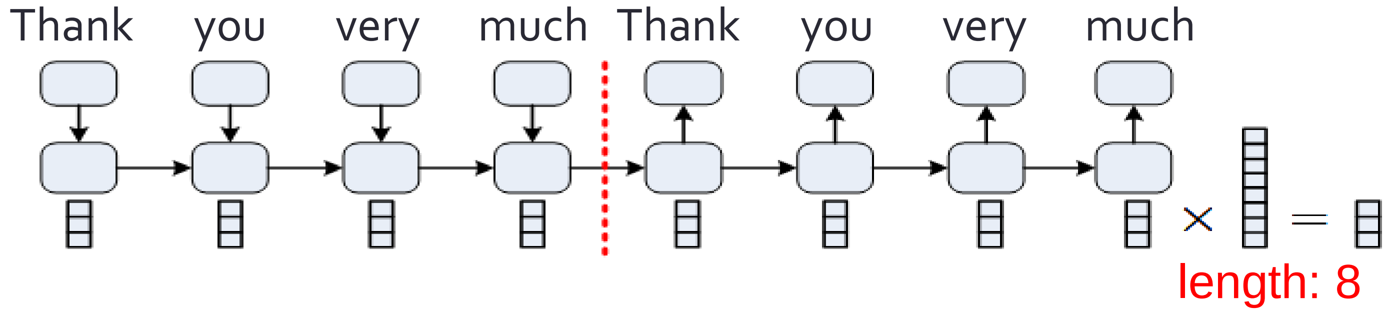
"Collaborative recurrent autoencoder: recommend while learning to fill in the blanks" [**Wang** et al., NIPS 2016a]

Challenges:

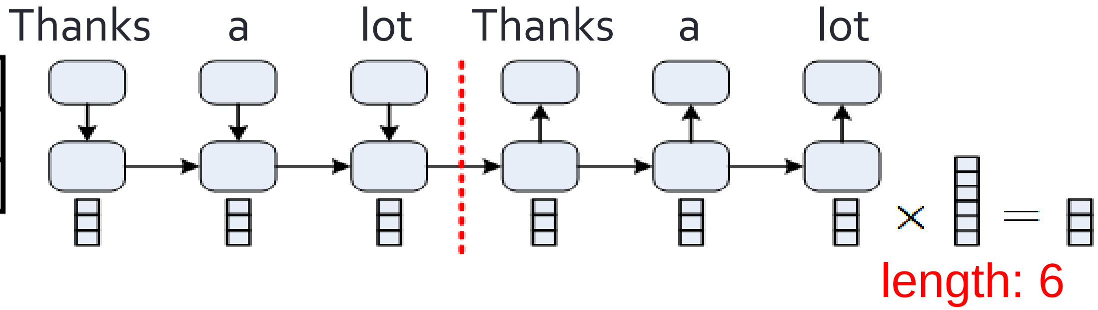
- RNN encoder may learn wrong transition between words
- Pool a **variable**-length sequence into a **fixed**-length vector

Challenge 2: Variable-Length Vector for Pooling

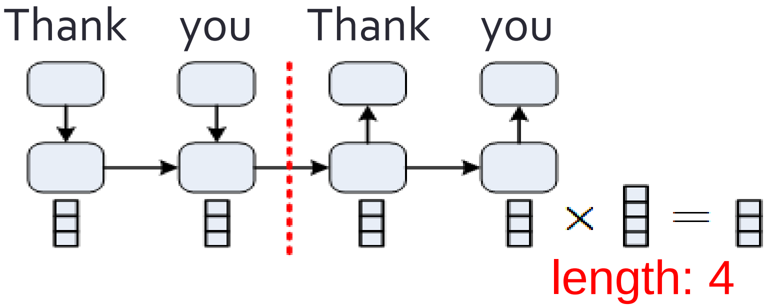
vector	length
sequence	4
weight	8



vector	length
sequence	3
weight	6

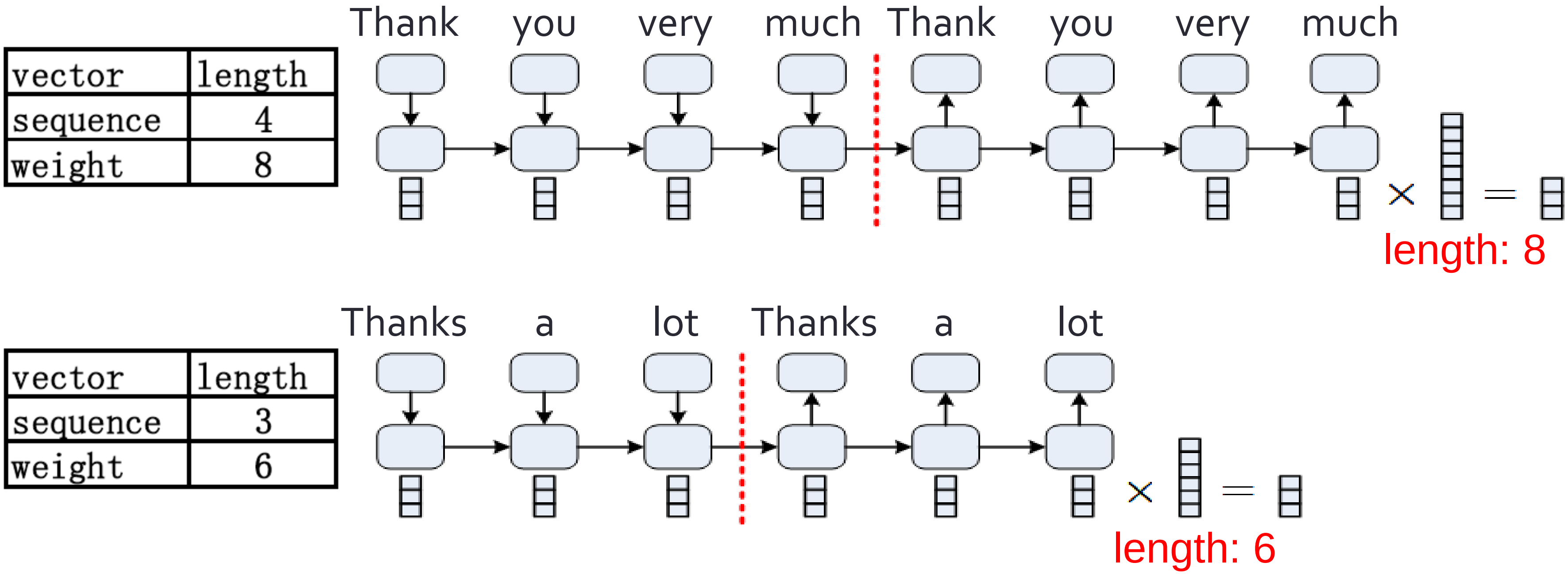


vector	length
sequence	2
weight	4



[**Wang** et al., NIPS 2016a]

Challenge 2: Variable-Length Vector for Pooling



vector	length
sequence	4
weight	8

vector	length
sequence	3
weight	6

Straight-forward approach averages or sums the vectors

But different words should have different weights!
➔ Need to learn a variable-length weight vector

[**Wang** et al., NIPS 2016a]

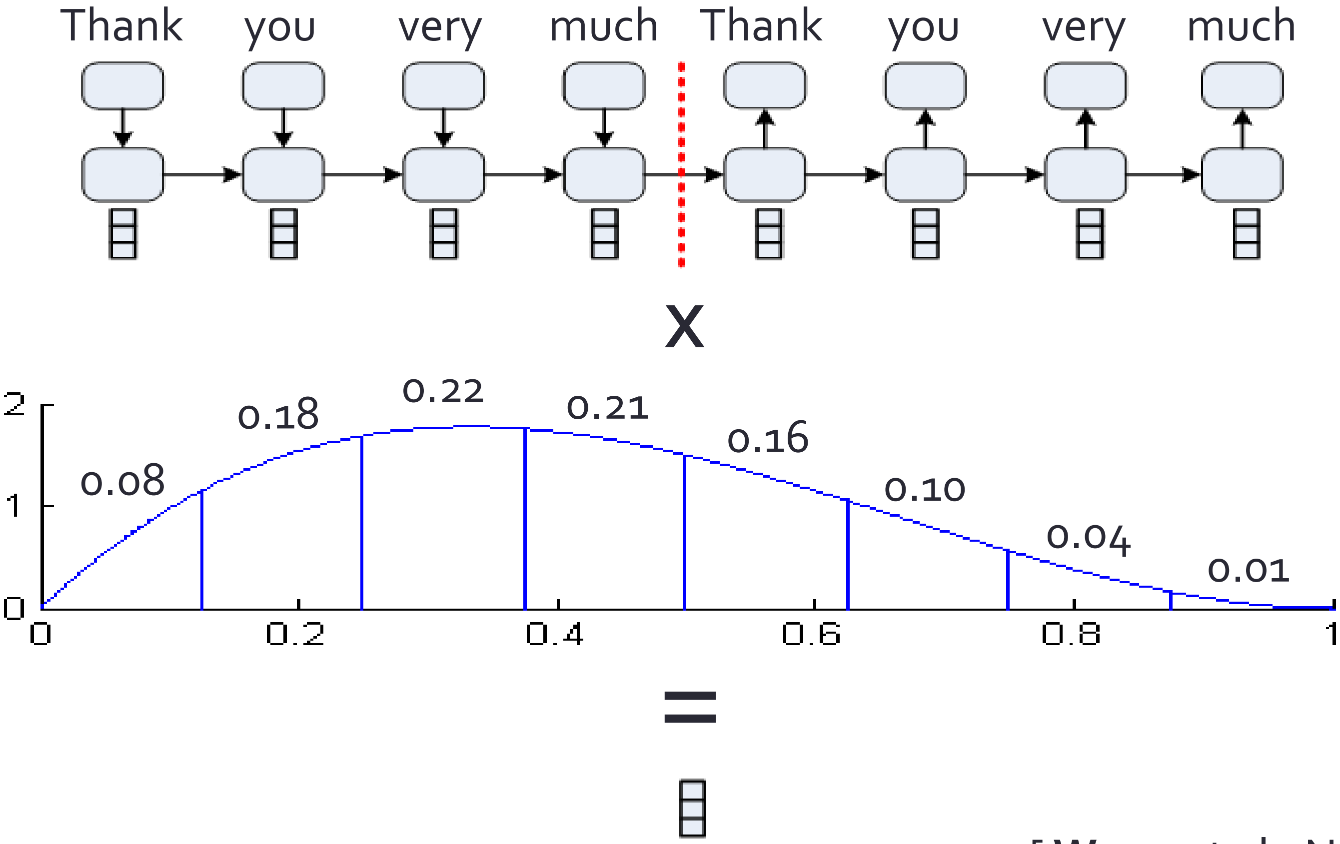
Challenge 2:
Variable-Length Weight Vector with Beta Distributions

Use the area of the beta distribution to define the weights!

8 vectors

length-8 weight vector

one single vector



[**Wang** et al., NIPS 2016a]

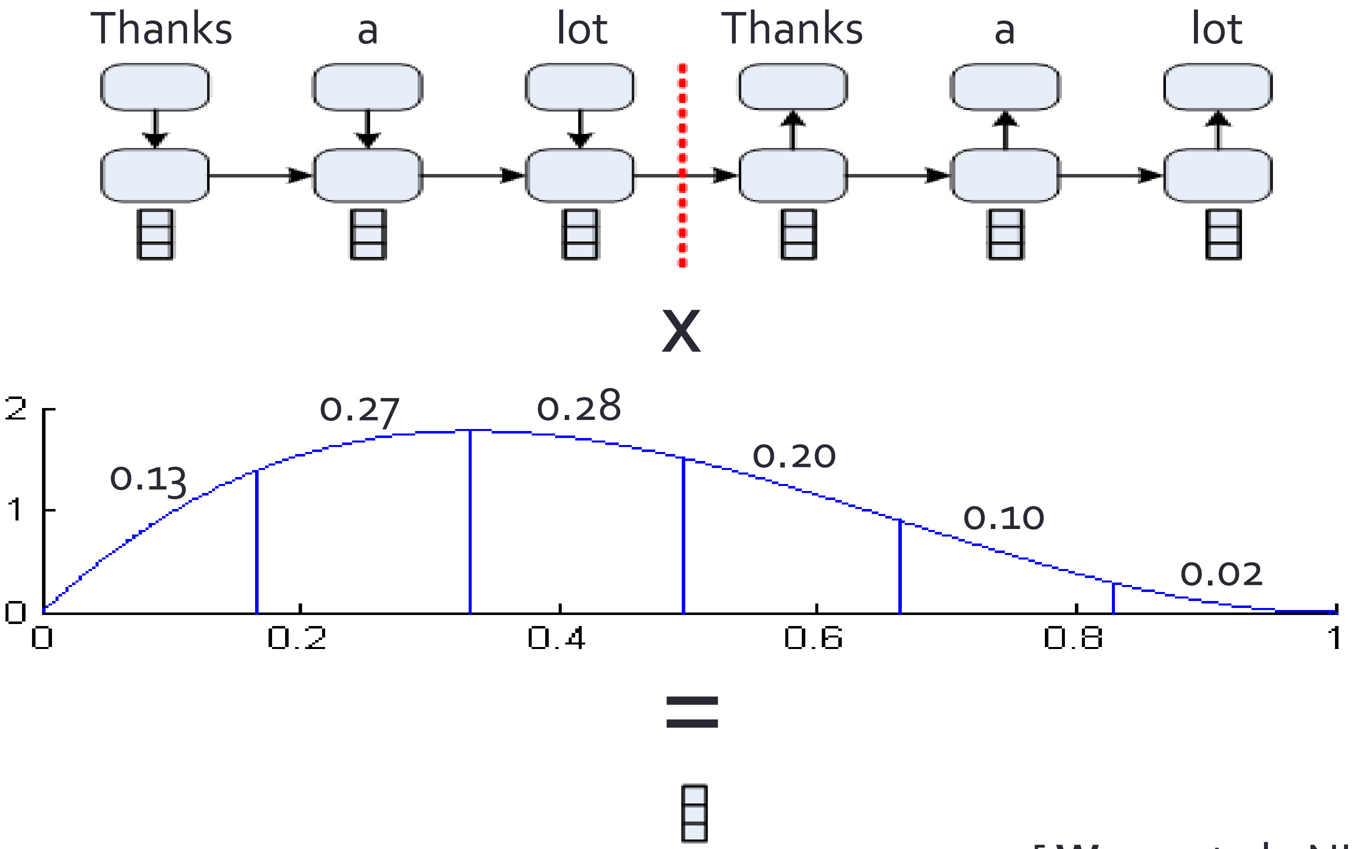
Challenge 2: Variable-Length Weight Vector with Beta Distributions

Use the area of the beta distribution to define the weights!

6 vectors

length-6 weight vector

one single vector

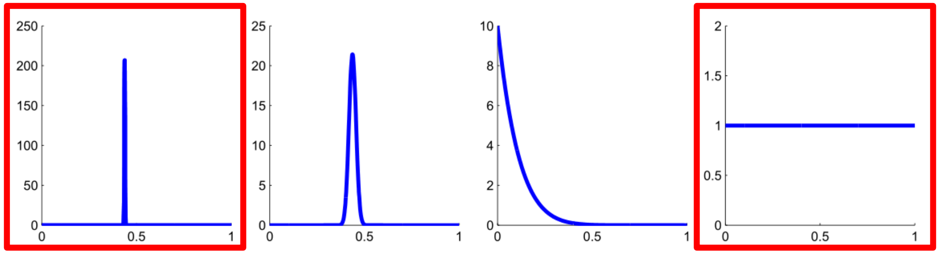


[**Wang** et al., NIPS 2016a]

Why Beta Distribution?

Because by learning two parameters a, b,
we can generate different variable-length weight vectors!

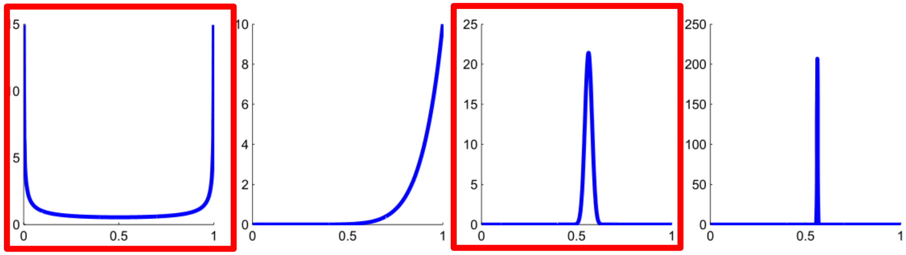
Beta distributions



Parameters

a	31112	311	1	1
b	40000	400	10	1
Recall	12.17	12.54	10.48	11.62

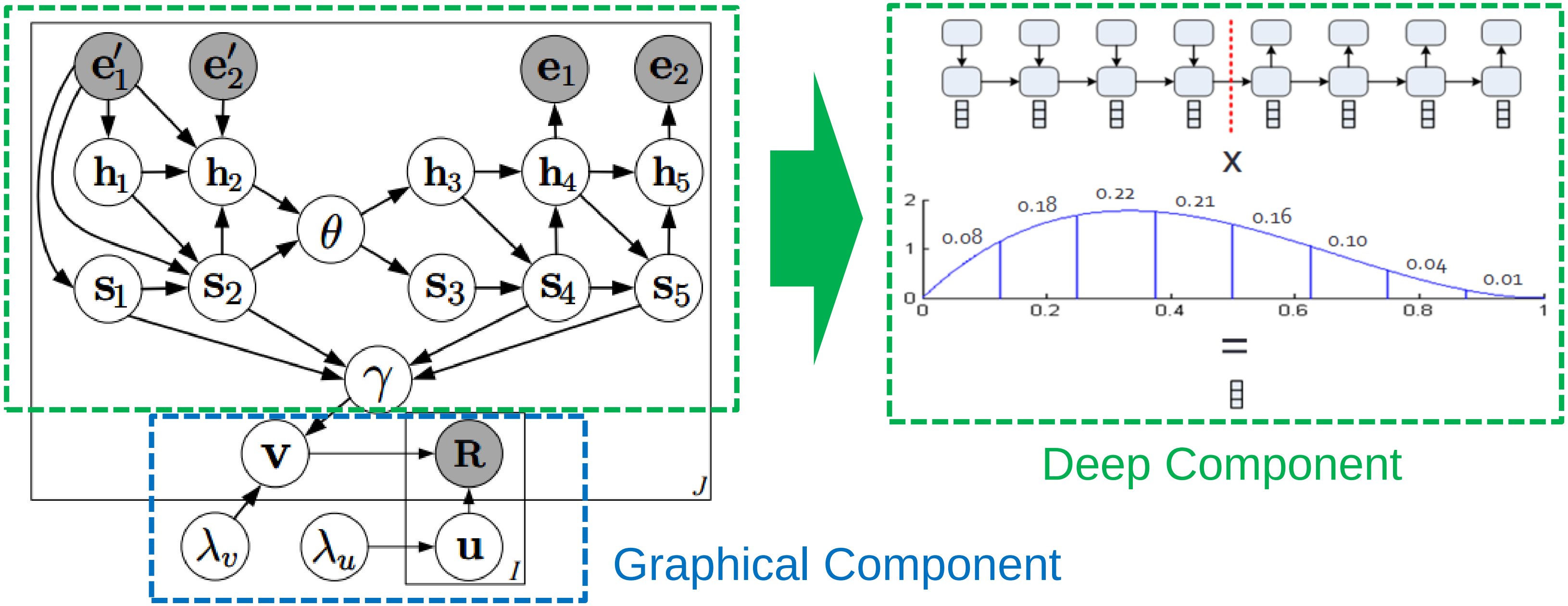
Beta distributions



Parameters

a	0.4	10	400	40000
b	0.4	1	311	31112
Recall	11.08	10.72	**12.71**	12.22

Overview: Current Model



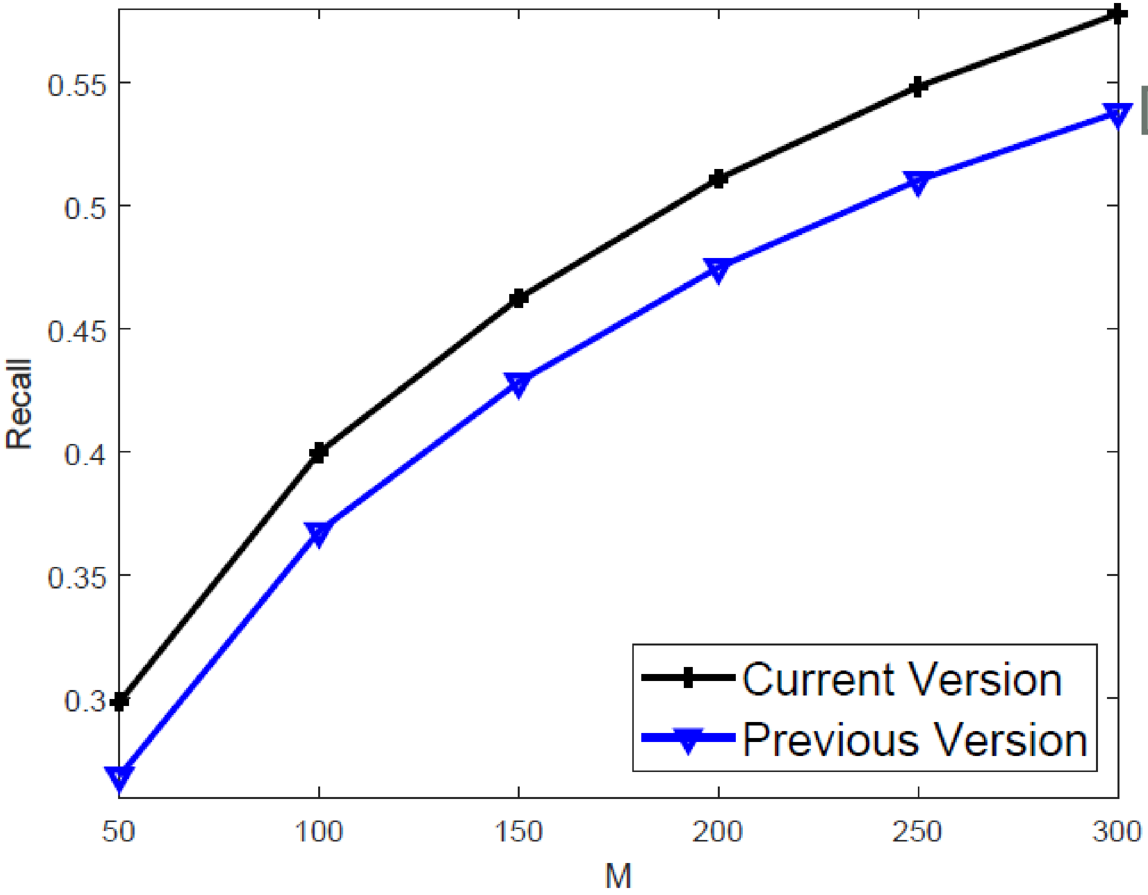
Deep Component

Graphical Component

- First model for joint recommendation and sequence generation
- Wildcard denoising for robust representation (**Challenge 1**)
- Beta-Pooling for variable-length sequences (**Challenge 2**)

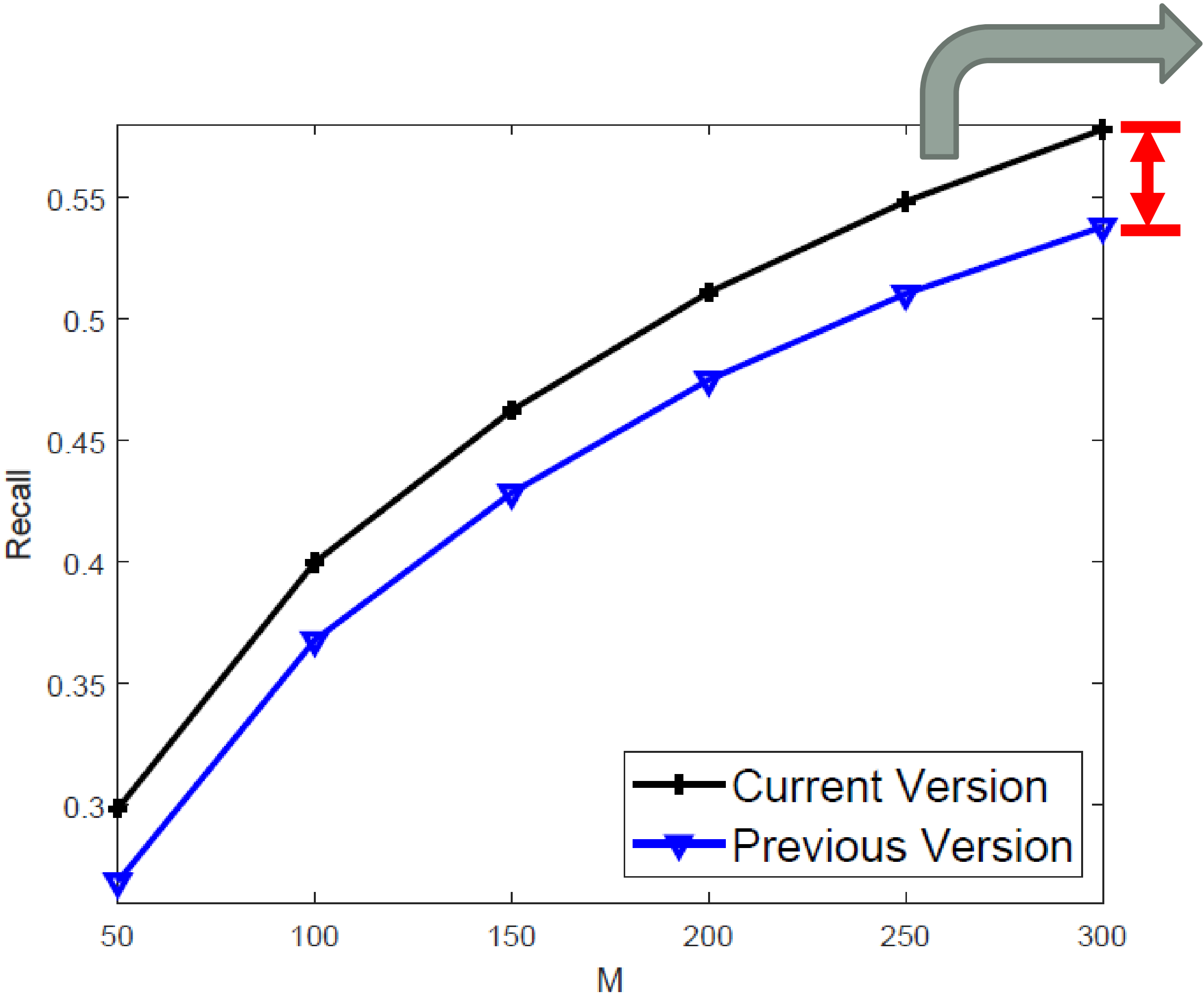
[**Wang** et al., NIPS 2016a]

Quantitative Comparison: Recall



Previous Version

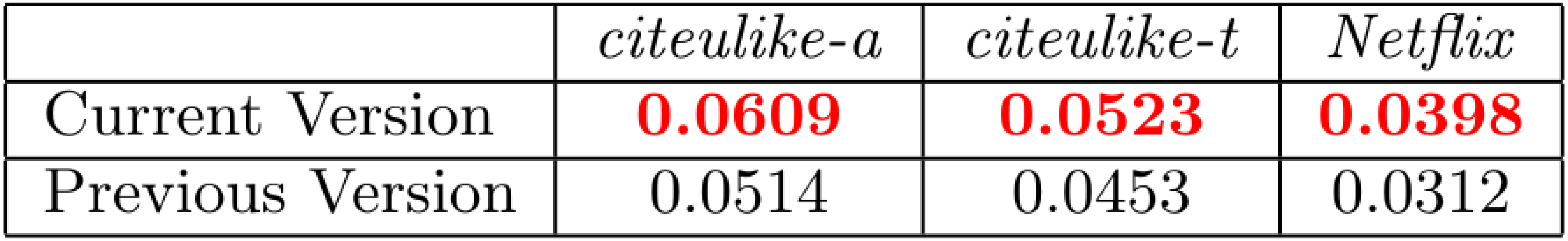
Quantitative Comparison: Recall



Current Version

Boost recall
from **54%** to **58%**

Quantitative Comparison: mAP



	citeulike-a	*citeulike-t*	*Netflix*
Current Version	**0.0609**	**0.0523**	**0.0398**
Previous Version	0.0514	0.0453	0.0312

Case Study: Comparing Previous and Current Version

Article User 1 Read	Bayesian adaptive user profiling with explicit and implicit feedback	

[**Wang** et al., NIPS 2016a]

Case Study: Comparing Previous and Current Version

Article User 1 Read	Bayesian adaptive user profiling with explicit and implicit feedback	
	CRAE (Current Method)	Correct?
Recommended Articles		

[**Wang** et al., NIPS 2016a]

Case Study: Comparing Previous and Current Version

Article User 1 Read	Bayesian adaptive user profiling with explicit and implicit feedback	
	CRAE (Current Method)	Correct?
Recommended Articles		
	CDL (Previous Method)	Correct?
Recommended Articles		

[**Wang** et al., NIPS 2016a]

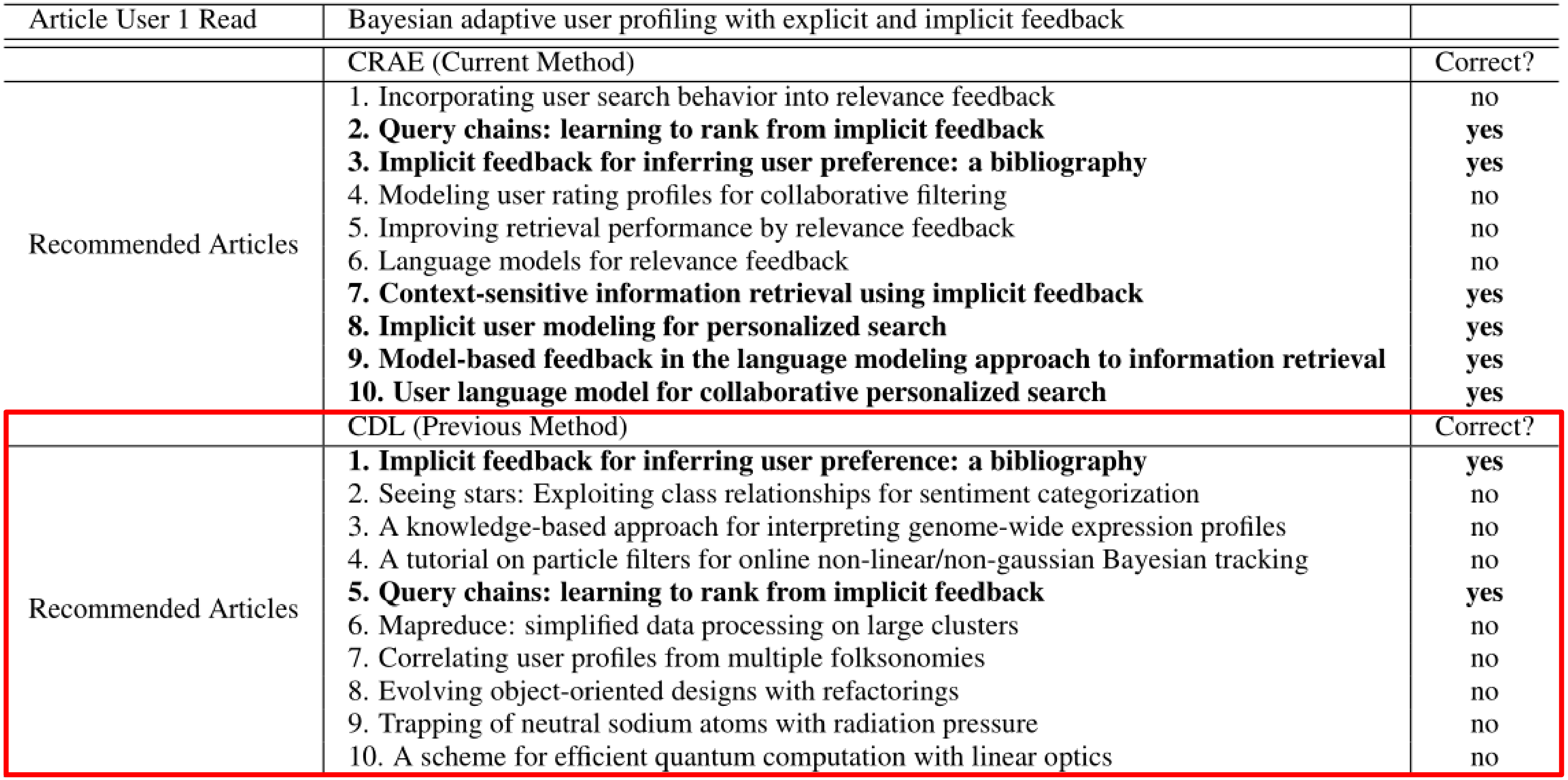
Case Study: Comparing Previous and Current Version

Article User 1 Read	Bayesian adaptive user profiling with explicit and implicit feedback	
	CRAE (Current Method)	Correct?
Recommended Articles	1. Incorporating user search behavior into relevance feedback	no
	2. Query chains: learning to rank from implicit feedback	**yes**
	3. Implicit feedback for inferring user preference: a bibliography	**yes**
	4. Modeling user rating profiles for collaborative filtering	no
	5. Improving retrieval performance by relevance feedback	no
	6. Language models for relevance feedback	no
	7. Context-sensitive information retrieval using implicit feedback	**yes**
	8. Implicit user modeling for personalized search	**yes**
	9. Model-based feedback in the language modeling approach to information retrieval	**yes**
	10. User language model for collaborative personalized search	**yes**
	CDL (Previous Method)	Correct?
Recommended Articles		

Precision: **60%**

[**Wang** et al., NIPS 2016a]

Case Study: Comparing Previous and Current Version



Article User 1 Read	Bayesian adaptive user profiling with explicit and implicit feedback	
	CRAE (Current Method)	Correct?
Recommended Articles	1. Incorporating user search behavior into relevance feedback	no
	2. Query chains: learning to rank from implicit feedback	**yes**
	3. Implicit feedback for inferring user preference: a bibliography	**yes**
	4. Modeling user rating profiles for collaborative filtering	no
	5. Improving retrieval performance by relevance feedback	no
	6. Language models for relevance feedback	no
	7. Context-sensitive information retrieval using implicit feedback	**yes**
	8. Implicit user modeling for personalized search	**yes**
	9. Model-based feedback in the language modeling approach to information retrieval	**yes**
	10. User language model for collaborative personalized search	**yes**
	CDL (Previous Method)	Correct?
Recommended Articles	**1. Implicit feedback for inferring user preference: a bibliography**	**yes**
	2. Seeing stars: Exploiting class relationships for sentiment categorization	no
	3. A knowledge-based approach for interpreting genome-wide expression profiles	no
	4. A tutorial on particle filters for online non-linear/non-gaussian Bayesian tracking	no
	5. Query chains: learning to rank from implicit feedback	**yes**
	6. Mapreduce: simplified data processing on large clusters	no
	7. Correlating user profiles from multiple folksonomies	no
	8. Evolving object-oriented designs with refactorings	no
	9. Trapping of neutral sodium atoms with radiation pressure	no
	10. A scheme for efficient quantum computation with linear optics	no

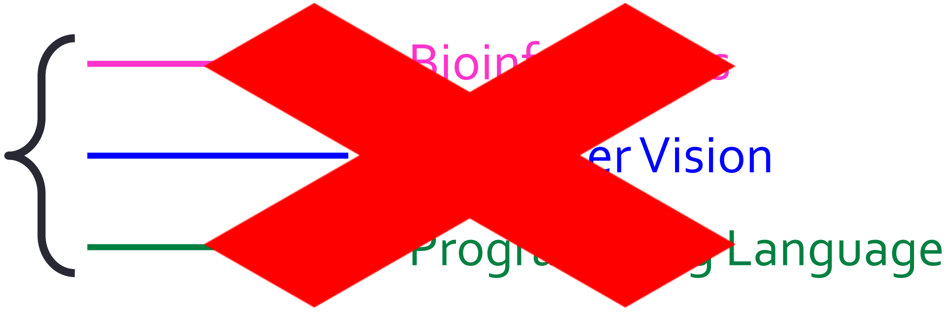
Precision: **60%** VS **20%**

[**Wang** et al., NIPS 2016a]

Results from Previous Version

User Profiling & Information Retrieval

Article User 1 Read	Bayesian adaptive user profiling with explicit and implicit feedback	
	CDL (Previous Method)	Correct?
Recommended Articles	**1. Implicit feedback for inferring user preference: a bibliography**	**yes**
	2. Seeing stars: Exploiting class relationships for sentiment categorization	no
	3. A knowledge-based approach for interpreting genome-wide expression profiles	no
	4. A tutorial on particle filters for online non-linear/non-gaussian Bayesian tracking	no
	5. Query chains: learning to rank from implicit feedback	**yes**
	6. Mapreduce: simplified data processing on large clusters	no
	7. Correlating user profiles from multiple folksonomies	no
	8. Evolving object-oriented designs with refactorings	no
	9. Trapping of neutral sodium atoms with radiation pressure	no
	10. A scheme for efficient quantum computation with linear optics	no



Incorrect Recommendations

Bioinf s

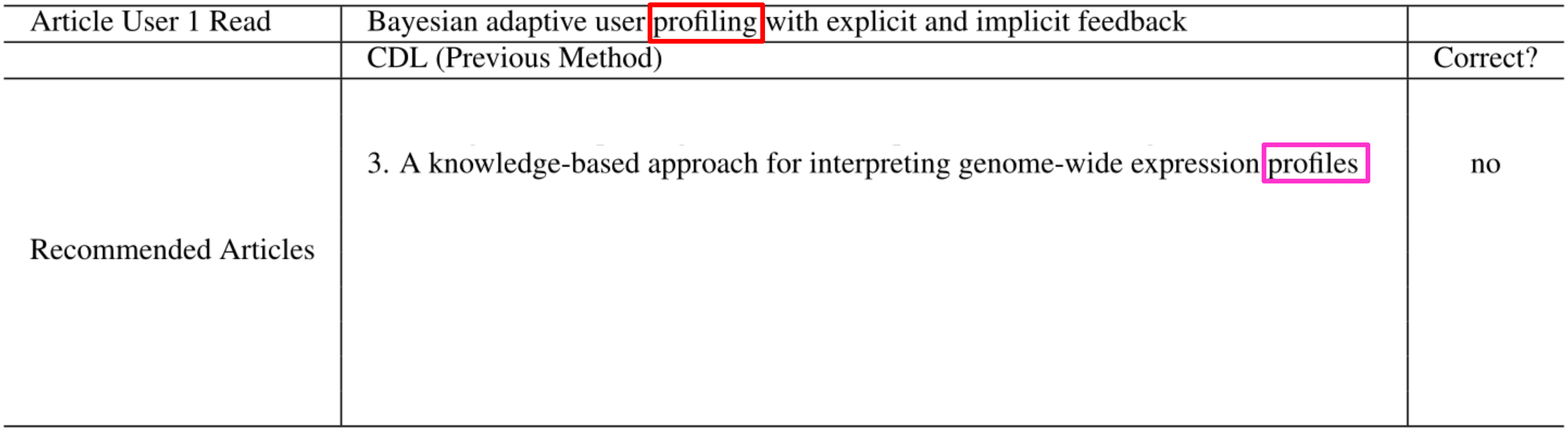
er Vision

Progr g Language

[**Wang** et al., NIPS 2016a]

Results from Previous Version

User Profiling & Information Retrieval

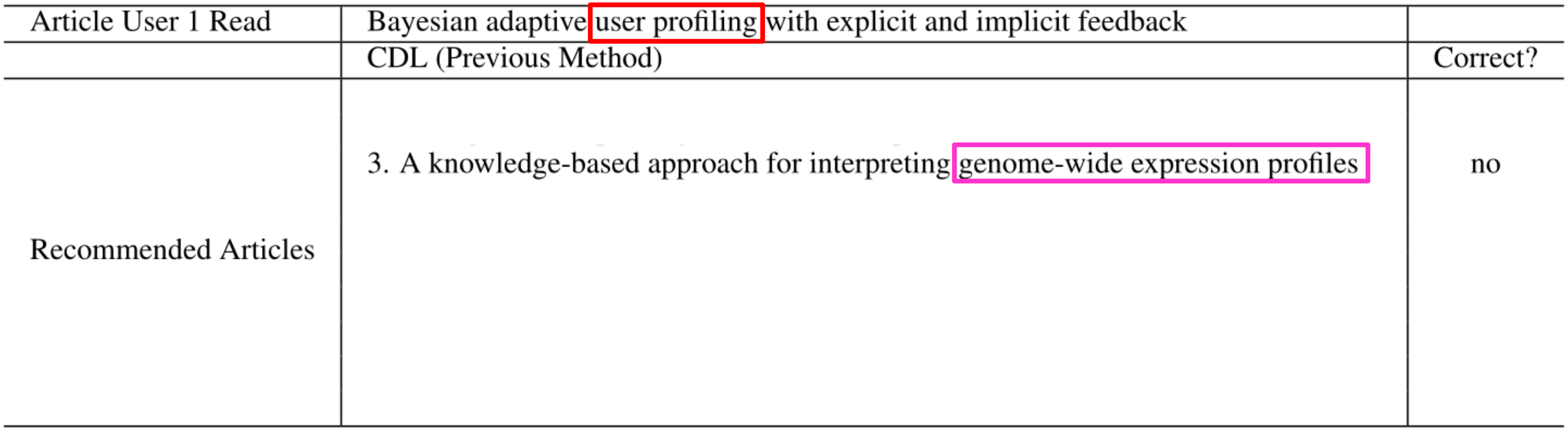


Article User 1 Read	Bayesian adaptive user profiling with explicit and implicit feedback	
	CDL (Previous Method)	Correct?
Recommended Articles	3. A knowledge-based approach for interpreting genome-wide expression profiles	no

Bioinformatics

Results from Previous Version

User Profiling & Information Retrieval



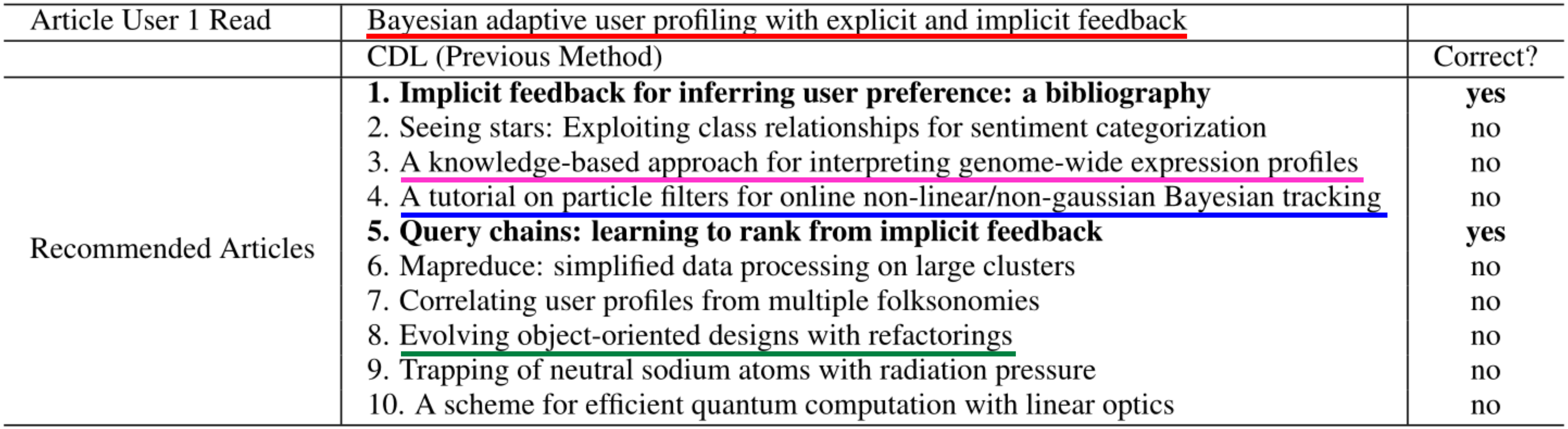
Article User 1 Read	Bayesian adaptive user profiling with explicit and implicit feedback	
	CDL (Previous Method)	Correct?
Recommended Articles	3. A knowledge-based approach for interpreting genome-wide expression profiles	no

Bioinformatics

They are very different articles!

Results from Previous Version

User Profiling & Information Retrieval



Article User 1 Read	Bayesian adaptive user profiling with explicit and implicit feedback	
	CDL (Previous Method)	Correct?
Recommended Articles	**1. Implicit feedback for inferring user preference: a bibliography**	**yes**
	2. Seeing stars: Exploiting class relationships for sentiment categorization	no
	3. A knowledge-based approach for interpreting genome-wide expression profiles	no
	4. A tutorial on particle filters for online non-linear/non-gaussian Bayesian tracking	no
	5. Query chains: learning to rank from implicit feedback	**yes**
	6. Mapreduce: simplified data processing on large clusters	no
	7. Correlating user profiles from multiple folksonomies	no
	8. Evolving object-oriented designs with refactorings	no
	9. Trapping of neutral sodium atoms with radiation pressure	no
	10. A scheme for efficient quantum computation with linear optics	no

The current version can avoid this using the sequential information among words!

[**Wang** et al., NIPS 2016a]

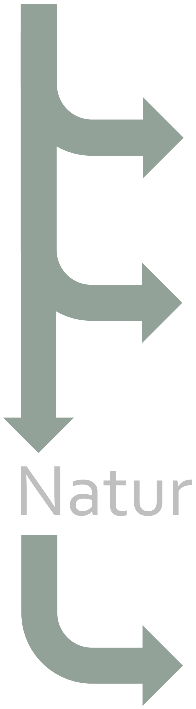
Contributions of BDL-Based Recommender Systems

First end-to-end recommender system that combines deep learning and graphical models

Robust probabilistic representation that deals with sequential text and missing words

Improve performance over the state of the art

Bayesian Deep Learning



A Unified Framework

Recommender Systems

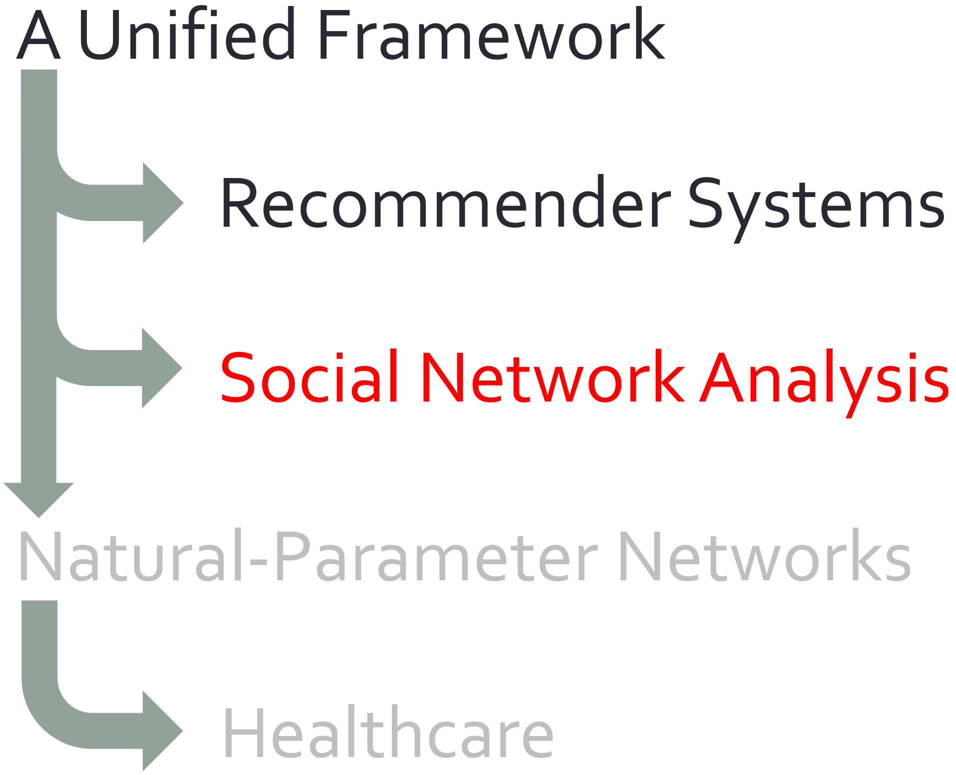
Social Network Analysis

Natural-Parameter Networks

Healthcare

[**Wang** et al., KDD 2015]
[**Wang** et al., NIPS 2016a]

Bayesian Deep Learning



A Unified Framework

Recommender Systems

Social Network Analysis

Natural-Parameter Networks

Healthcare

[**Wang** et al., AAAI 2015]
[**Wang** et al., AAAI 2017]
[**Huang, Xue, Wang, Wang.,** ICML 2020]

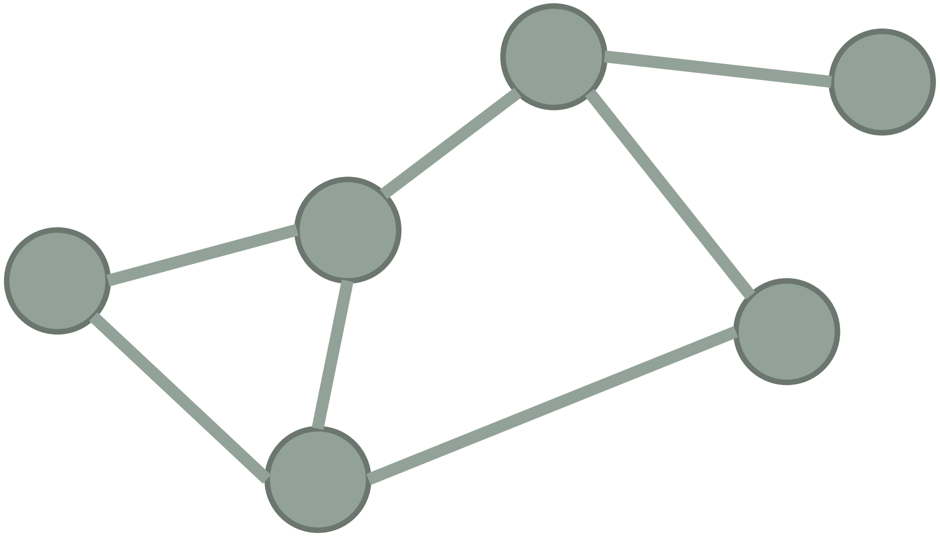
Problem:

Network (graph)

Relations between nodes

Node

Node content



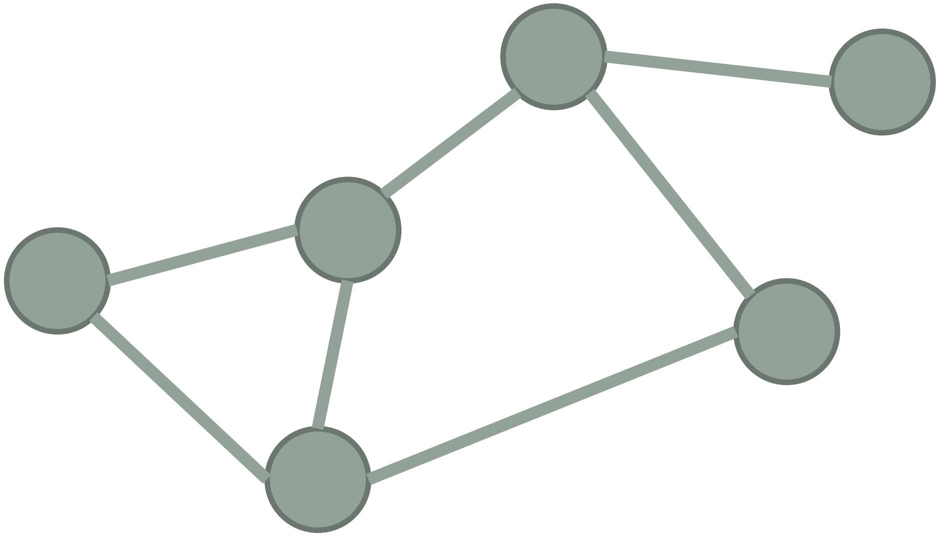
Problem:

Social Network

Friend relations

Node

Image, text, etc.



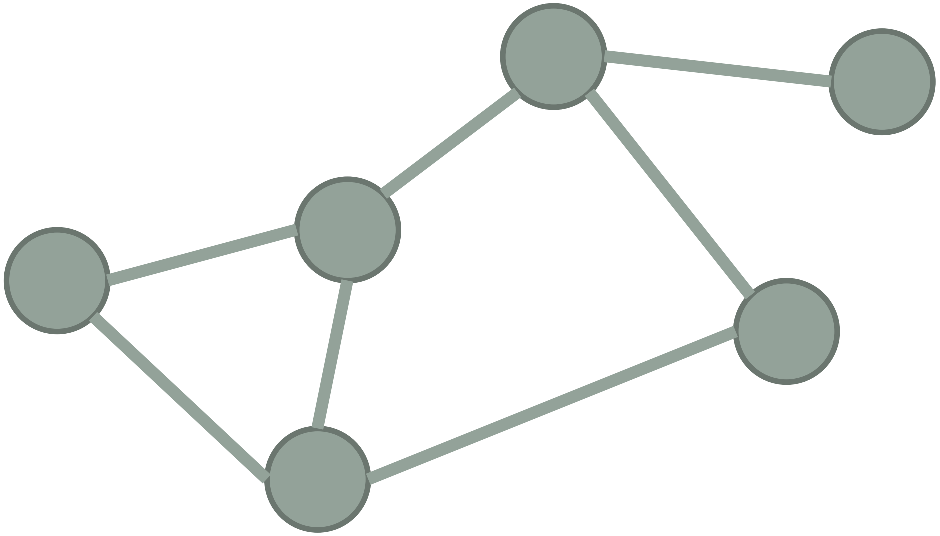
Problem: Learn a per-node representation that captures both content and graph

Citation Network

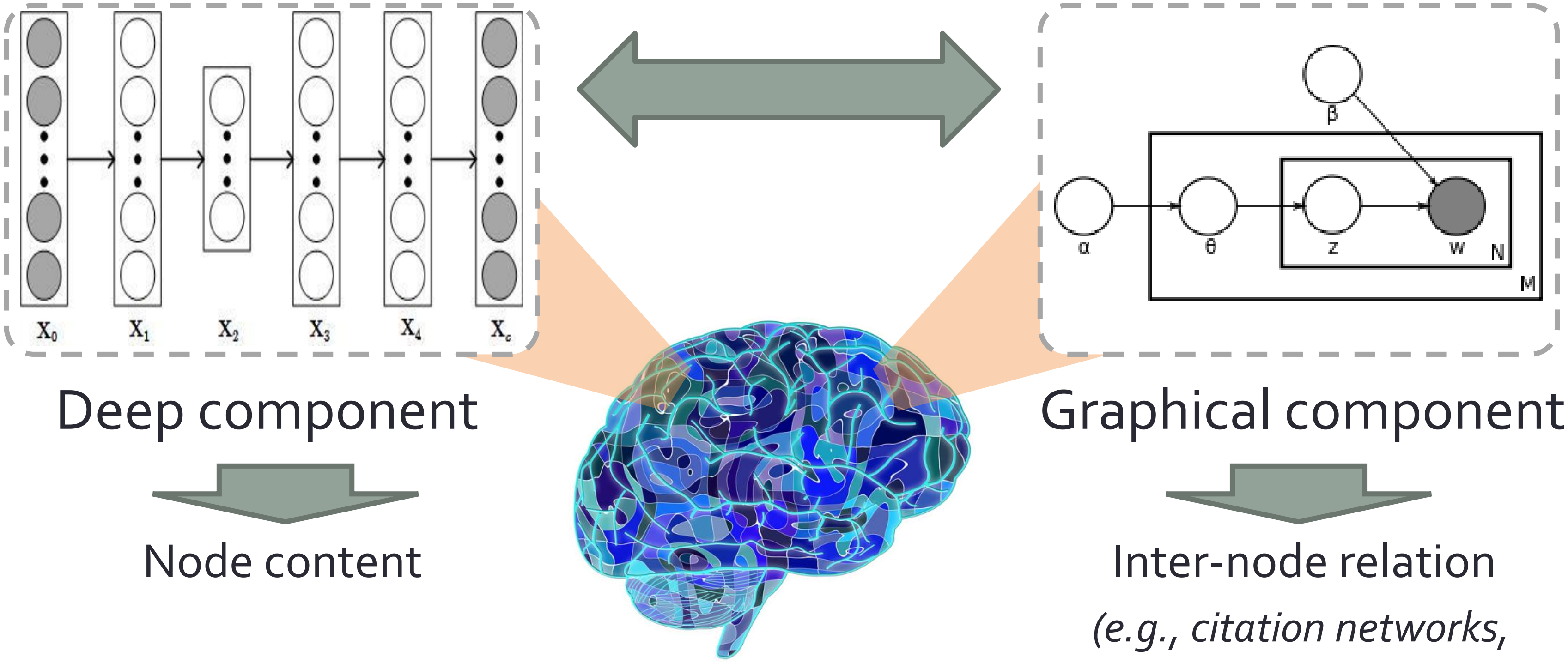
'Cited by' relations

Node

Article text



Solution: Relational Probabilistic Autoencoder



Deep component

Node content

Graphical component

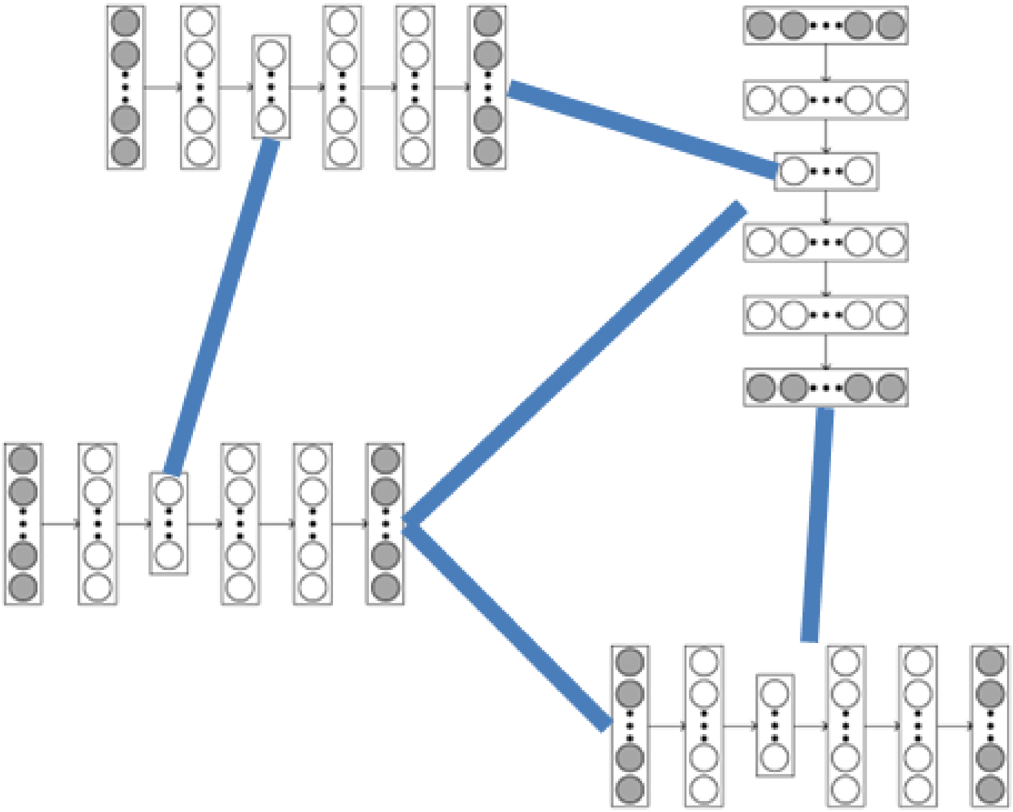
Inter-node relation

(e.g., citation networks, knowledge graphs)

BDL-Based Topic Models

[**Wang** et al. AAAI 2015]

Solution: Relational Probabilistic Autoencoder



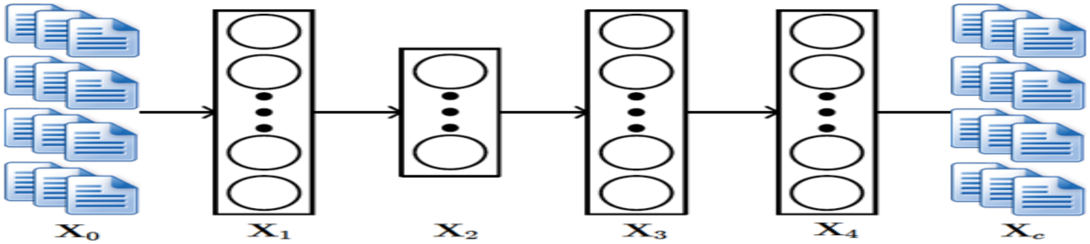
- Enhance representation power with relational information

Challenges

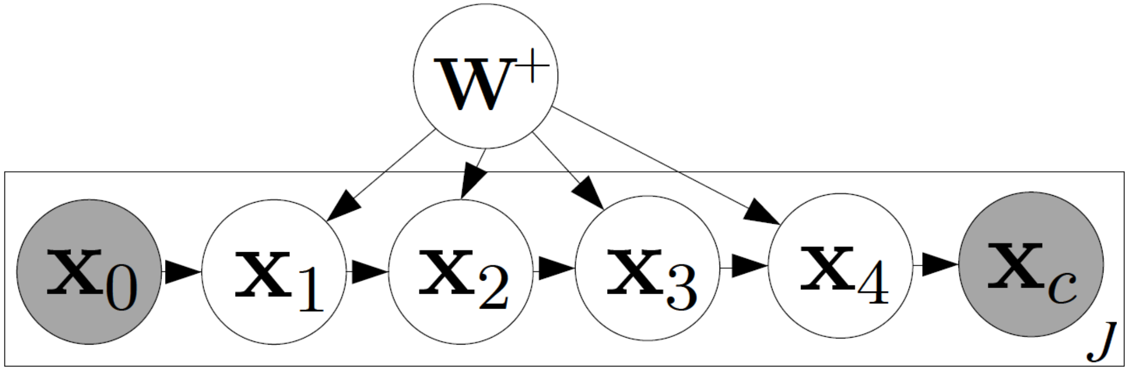
- How to make the representation of two nodes closer to each other if they are connected in the graph
- How to handle multiple graphs

Challenge 1: Representation of Connected Nodes

Standard autoencoder:



Probabilistic autoencoder:

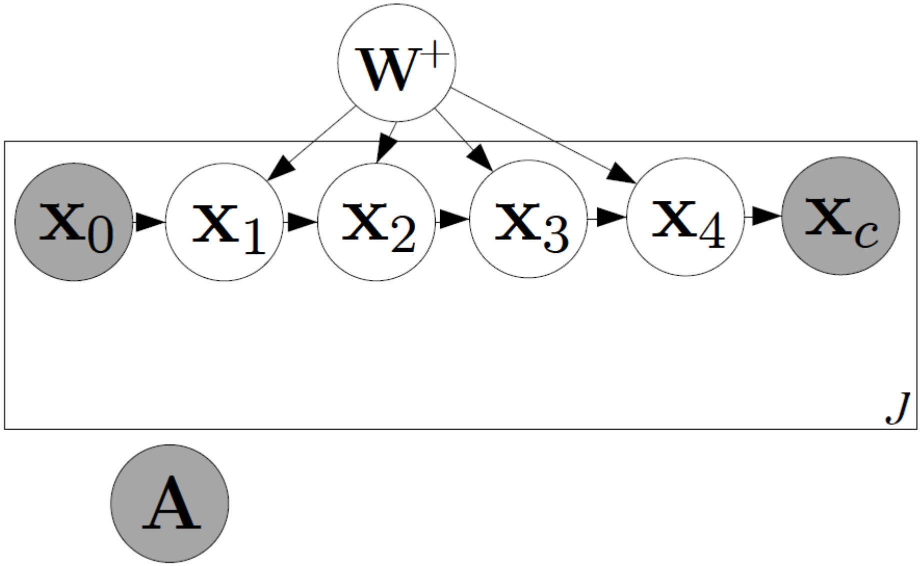


$$\mathbf{X}_{l,j*} \sim \mathcal{N}(\sigma(\mathbf{X}_{l-1,j*}\mathbf{W}_l + \mathbf{b}_l), \boxed{\lambda_s^{-1}\mathbf{I}_{K_l}})$$

Probabilistic Autoencoder: Gaussian noise after each nonlinear transformation

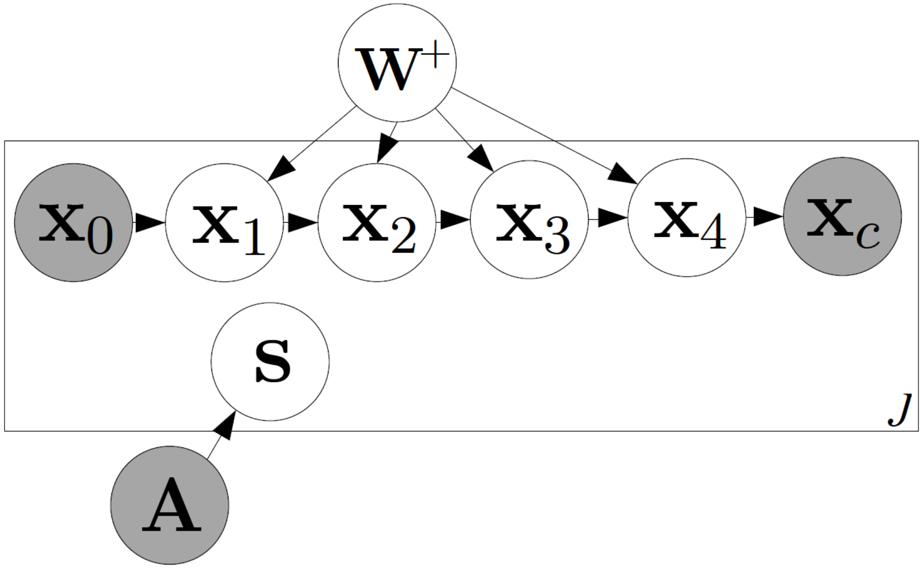
[**Wang** et al. KDD 2015]

Challenge 1
Step 1 of 3: Start from Adjacency Matrix

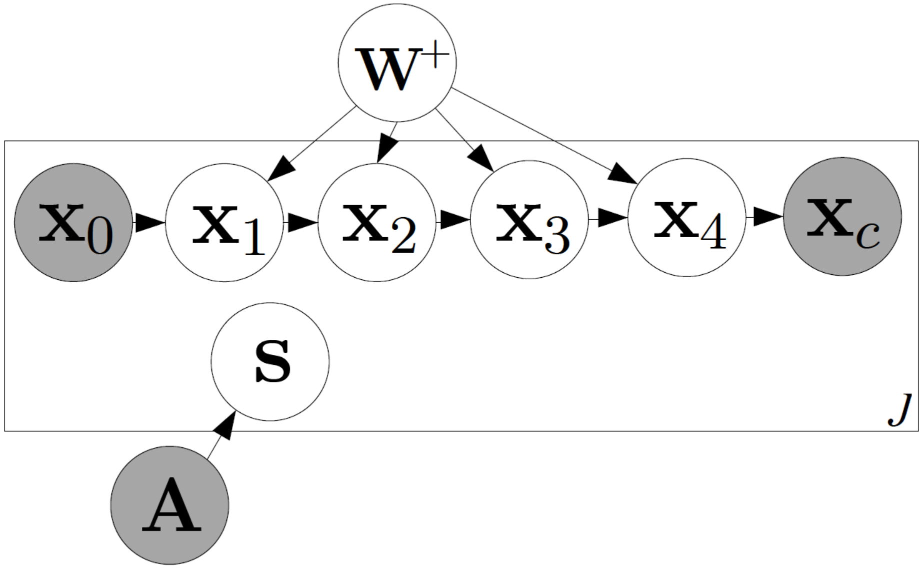


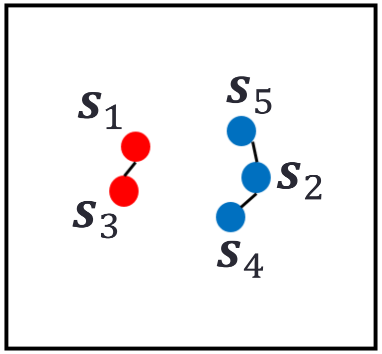
- $\boldsymbol{A}$: Adjacency matrix that defines the graph
- J: Number of nodes

Challenge 1
Step 2 of 3: Generate Latent Vectors for J items



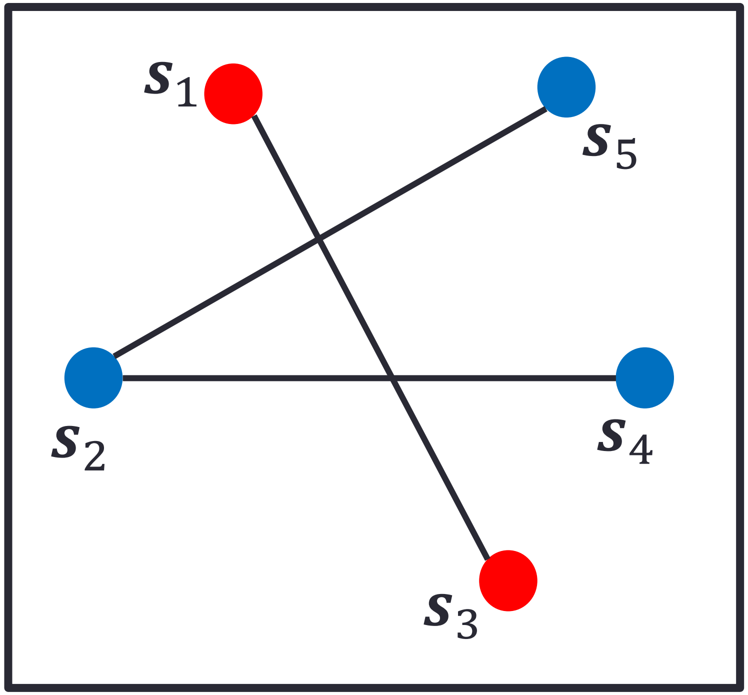
Challenge 1
Step 2 of 3: Generate Latent Vectors for J items

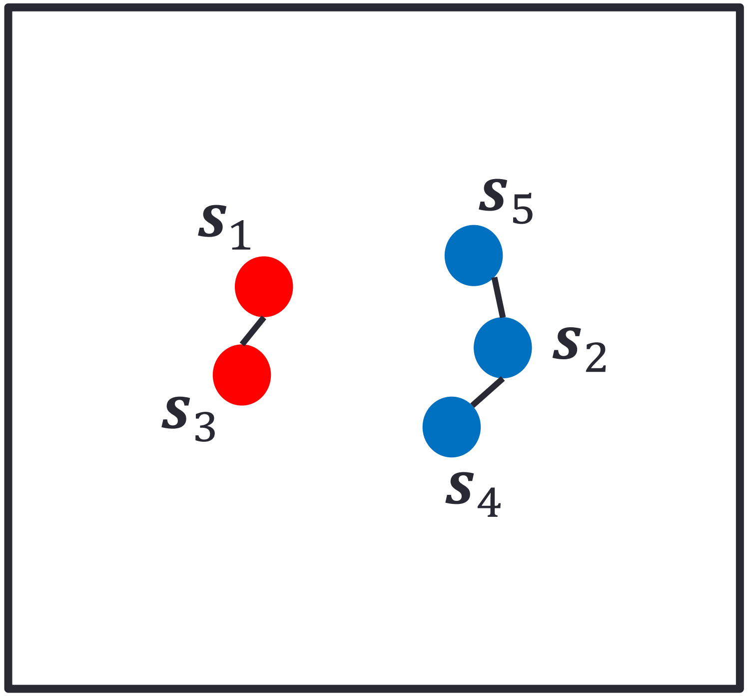




Ideally: Connected items closer to each other

Challenge 1
Step 2 of 3: Generate Latent Vectors for J items





Generate $\boldsymbol{s}_j$ one by one

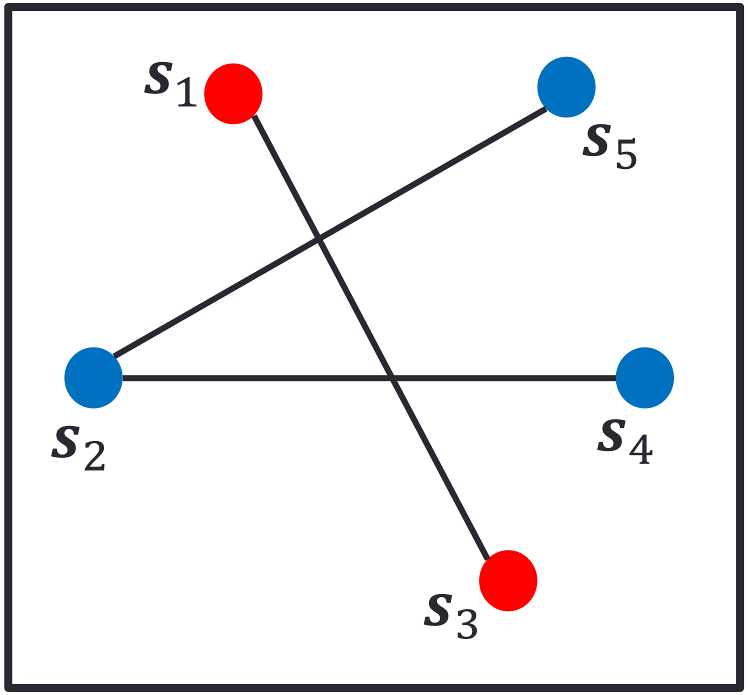
Generate $\boldsymbol{S} = [\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_J]$ all at once

Standard Gaussian $\mathbf{s}_j \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

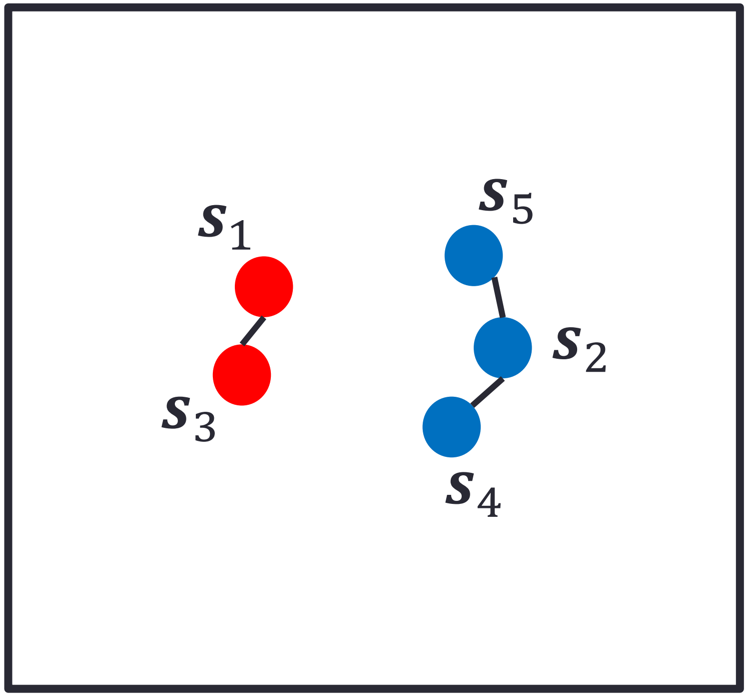
Matrix Gaussian distribution

$$p(\mathbf{S}) \propto \exp\{\mathrm{tr}[-\frac{\lambda_l}{2}\underline{\mathbf{S}\mathscr{L}_a}\mathbf{S}^T]\}$$

Challenge 1
Step 2 of 3: Generate Latent Vectors for J items



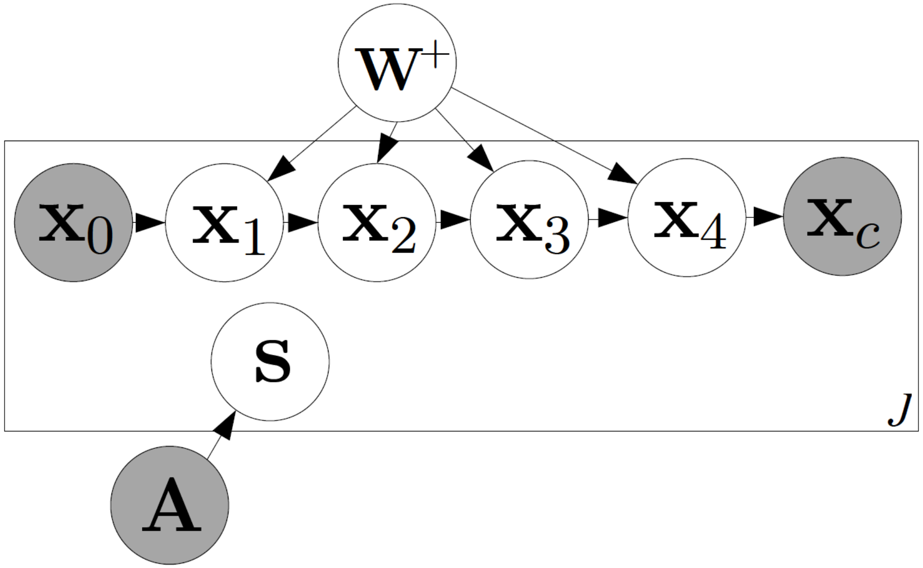
Low probability density



High probability density

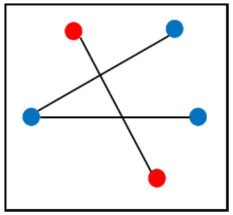
Matrix Gaussian distribution $p(\mathbf{S}) \propto \exp\{\mathrm{tr}[-\frac{\lambda_l}{2}\mathbf{S}\mathscr{L}_a\mathbf{S}^T]\}$

Challenge 1
Step 2 of 3: Generate Latent Vectors for J items

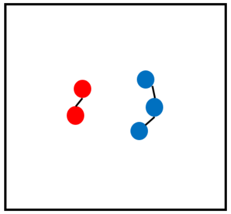


Generate **all *J* vectors** $\boldsymbol{S} = [s_1, s_2, \dots, s_J]$ from the **matrix-variate** Gaussian distribution:

$$p(\mathbf{S}) \propto \exp\{\mathrm{tr}[-\frac{\lambda_l}{2}\mathbf{S}\mathscr{L}_a\mathbf{S}^T]\}$$

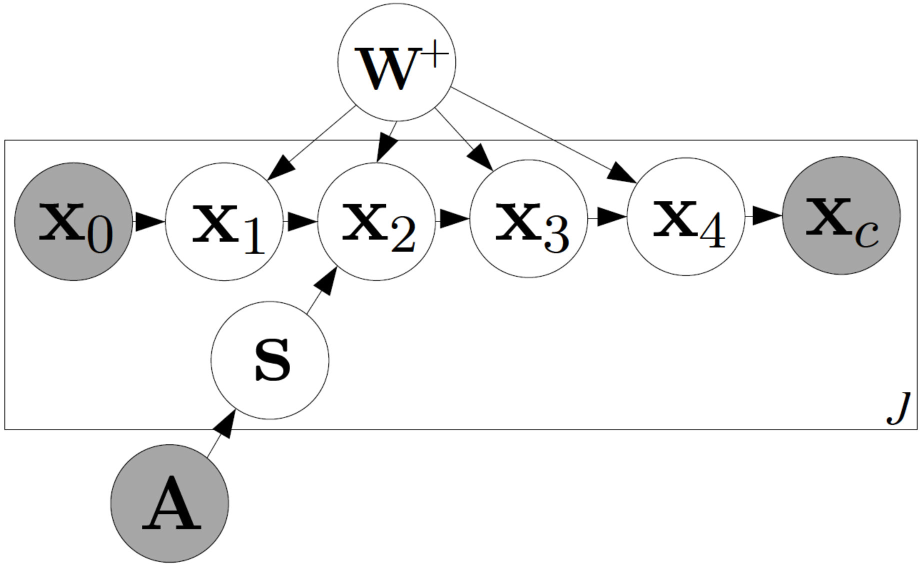


Standard Gaussian



$$p(\mathbf{S}) \propto \exp\{\mathrm{tr}[-\frac{\lambda_l}{2}\mathbf{S}\mathscr{L}_a\mathbf{S}^T]\}$$

Challenge 1
Step 3 of 3: Connect Latent Vectors to Representation



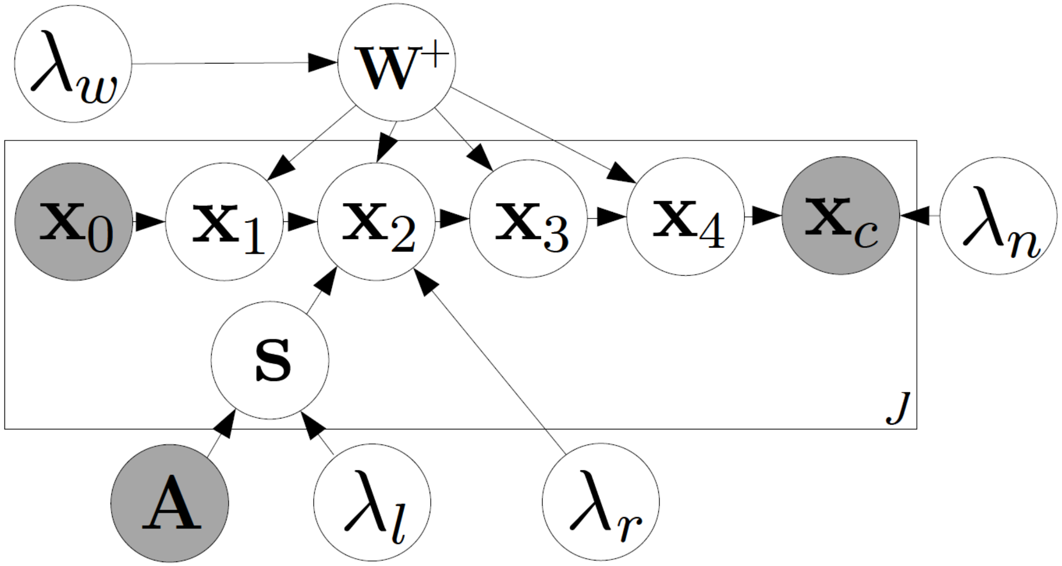
Generate middle-layer representation x_2 from Product of Gaussians (PoG):

$$\mathbf{x}_2 \sim \mathrm{PoG}(\mathbf{x}_1, \mathbf{s}_j)$$

First Gaussian related to x_1
Second Gaussian related to s_j

x_2 has information on both the **documents** x_0 and the **graph** A

Overview: Relational Autoencoder

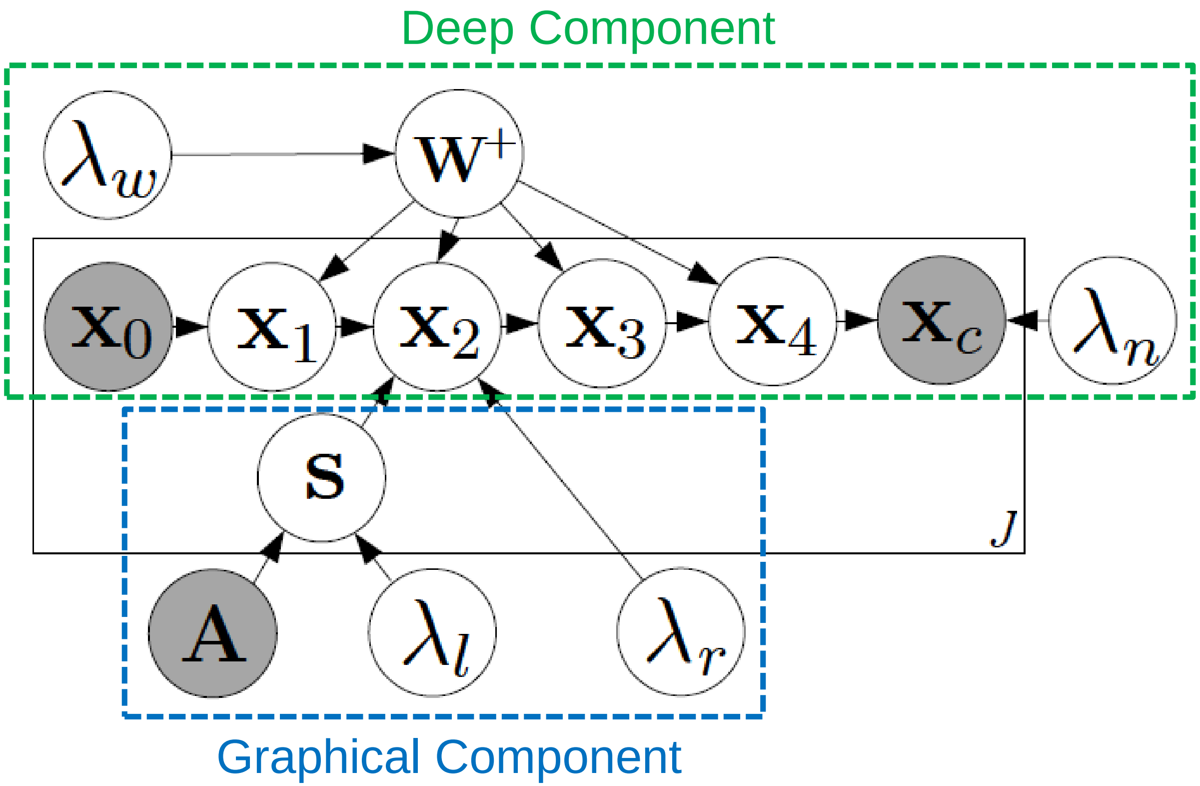


λ_w, λ_n, λ_l, λ_r: **hyperparamters** to control the **variance** of Gaussian distributions

Two key ingredients:

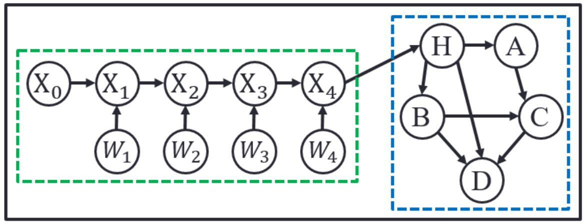
- Relational latent matrix $\boldsymbol{S}$ to represent $\boldsymbol{A}$
- PoG to connect $\boldsymbol{S}$, $\boldsymbol{X}_1$, and $\boldsymbol{X}_2$.

Relational Autoencoder: Two Components

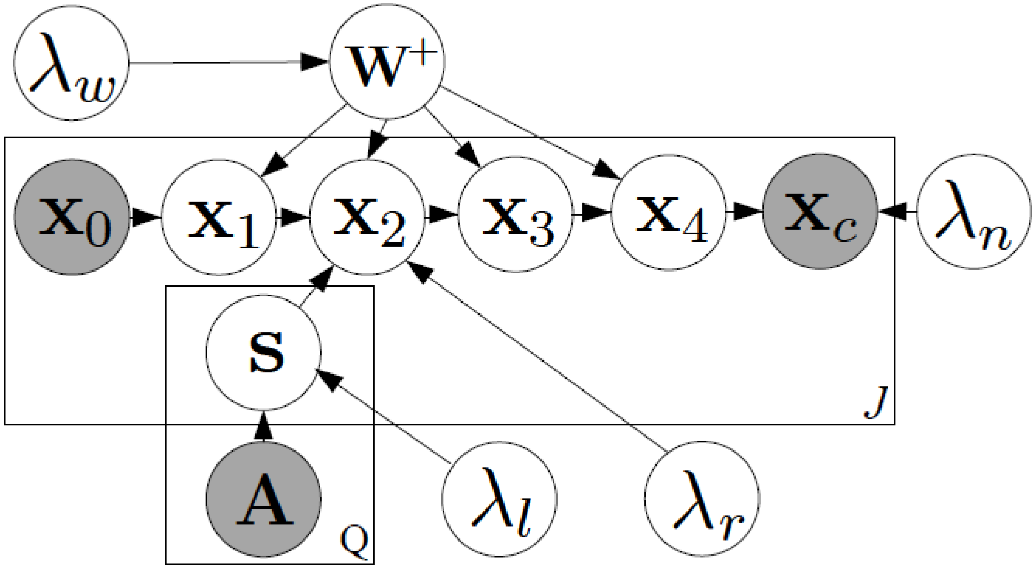


Deep Component

Graphical Component



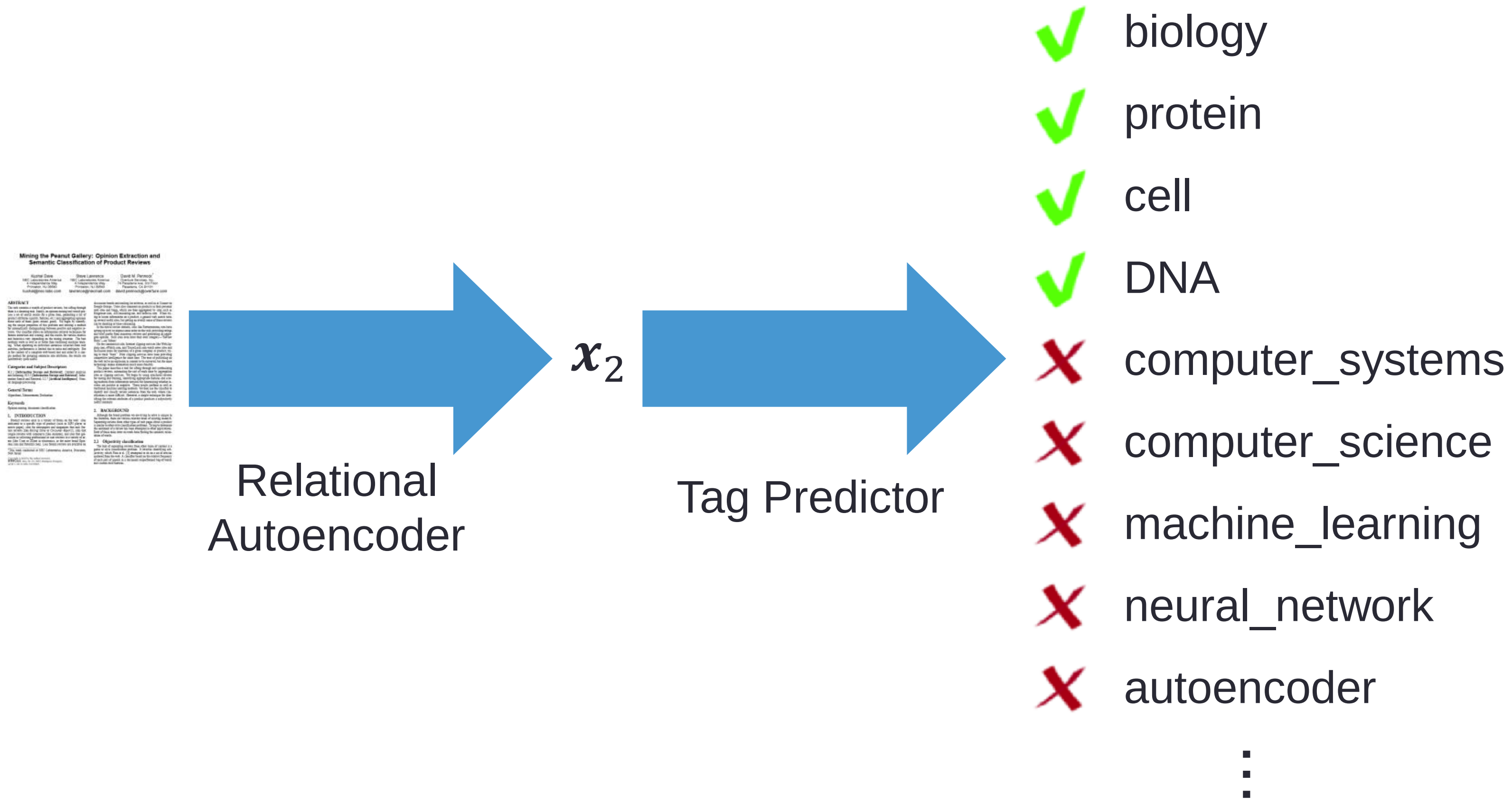
Challenge 2: Multiple Graphs (Networks)



Product of Q+1 Gaussians: $\mathbf{x}_2 \sim \mathrm{PoG}(\mathbf{x}_1, \mathbf{s}_j^{(1)}, \mathbf{s}_j^{(2)}, \ldots, \mathbf{s}_j^{(Q)})$

Multiple networks:
citation networks
co-author networks

Application: Predicting Tags for Articles



Mining the Peanut Gallery: Opinion Extraction and Semantic Classification of Product Reviews

Relational Autoencoder

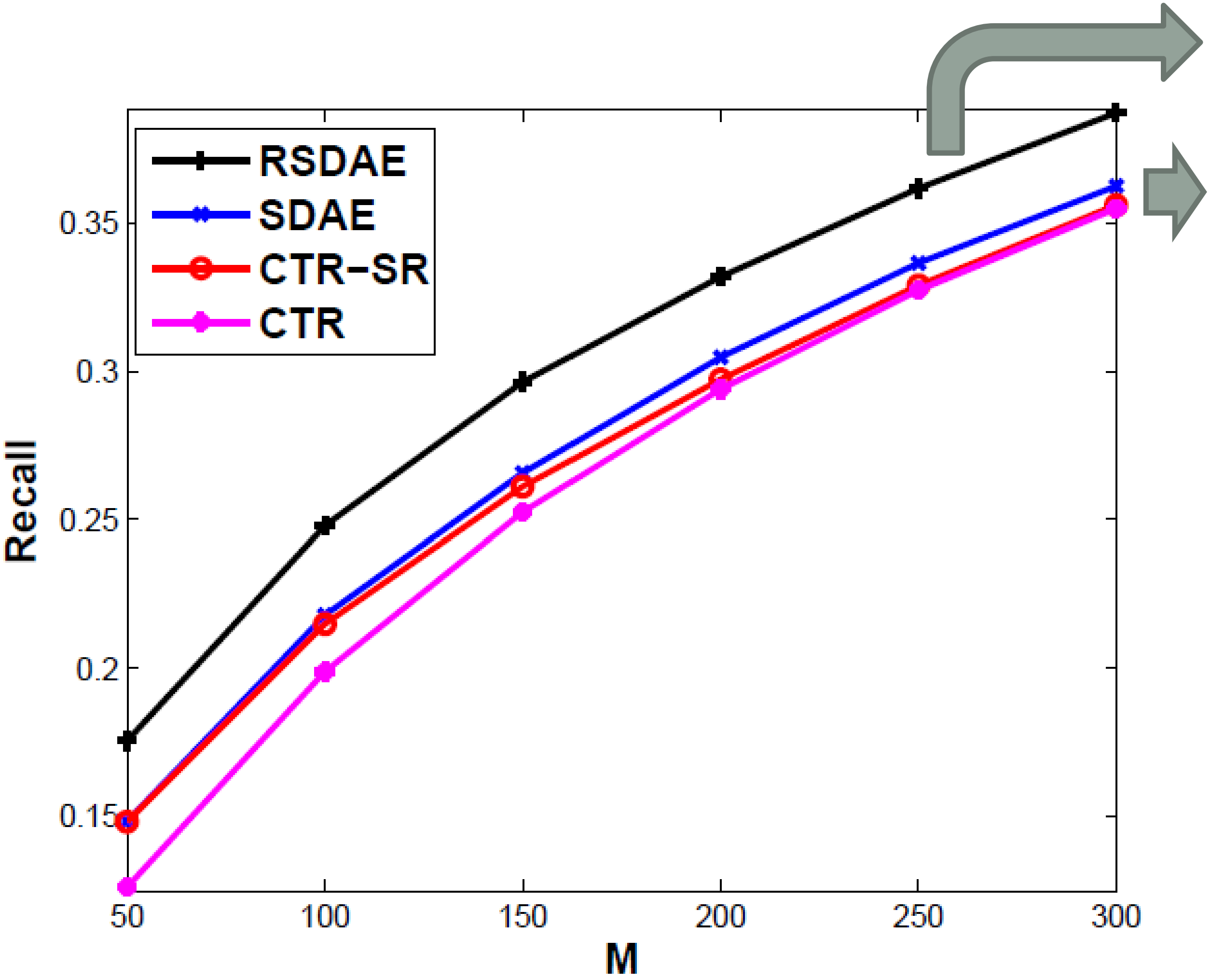
x_2

Tag Predictor

- ✓ biology
- ✓ protein
- ✓ cell
- ✓ DNA
- ✗ computer_systems
- ✗ computer_science
- ✗ machine_learning
- ✗ neural_network
- ✗ autoencoder

⋮

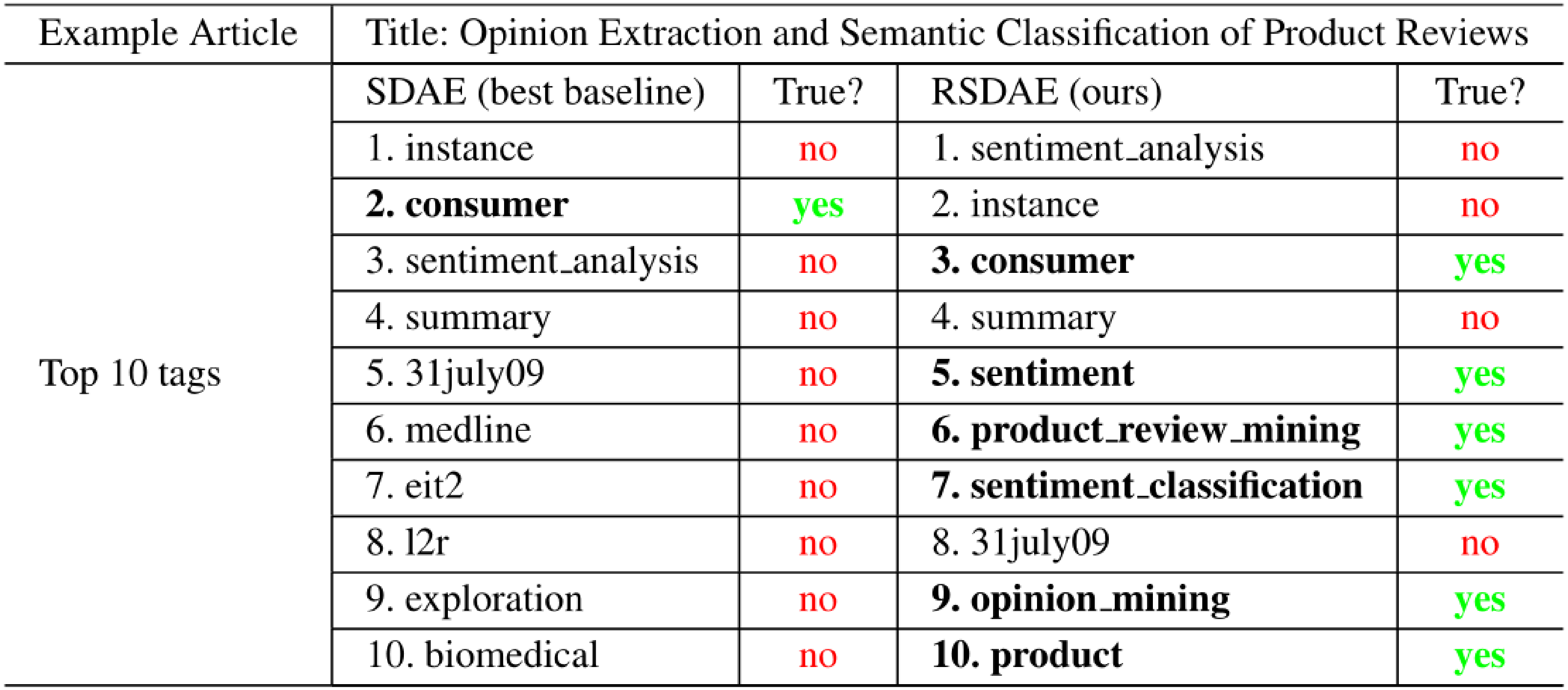
Sparse Setting, *citeulike-a*



Our method: **RSDAE**

Autoencoder (SDAE) outperforms CTR-SR

Case Study 1: Tagging Scientific Articles



Example Article	Title: Opinion Extraction and Semantic Classification of Product Reviews			
	SDAE (best baseline)	True?	RSDAE (ours)	True?
Top 10 tags	1. instance	no	1. sentiment_analysis	no
	2. consumer	yes	2. instance	no
	3. sentiment_analysis	no	**3. consumer**	yes
	4. summary	no	4. summary	no
	5. 31july09	no	**5. sentiment**	yes
	6. medline	no	**6. product_review_mining**	yes
	7. eit2	no	**7. sentiment_classification**	yes
	8. l2r	no	8. 31july09	no
	9. exploration	no	**9. opinion_mining**	yes
	10. biomedical	no	**10. product**	yes

Precision: 10% VS 60%

Case Study 2: Tagging Movies (Baseline)

Example Movie	Title: E.T. the Extra-Terrestrial	
	SDAE (best baseline)	True?
Top 10 tags	**1. Saturn Award (Best Special Effects)**	yes
	2. Want	no
	3. Saturn Award (Best Fantasy Film)	no
	4. Saturn Award (Best Writing)	yes
	5. Cool but freaky	no
	6. Saturn Award (Best Director)	no
	7. Oscar (Best Editing)	no
	8. almost favorite	no
	9. Steven Spielberg	yes
	10. sequel better than original	no

Precision: 30% VS 60%

Case Study 2: Tagging Movies (Ours)

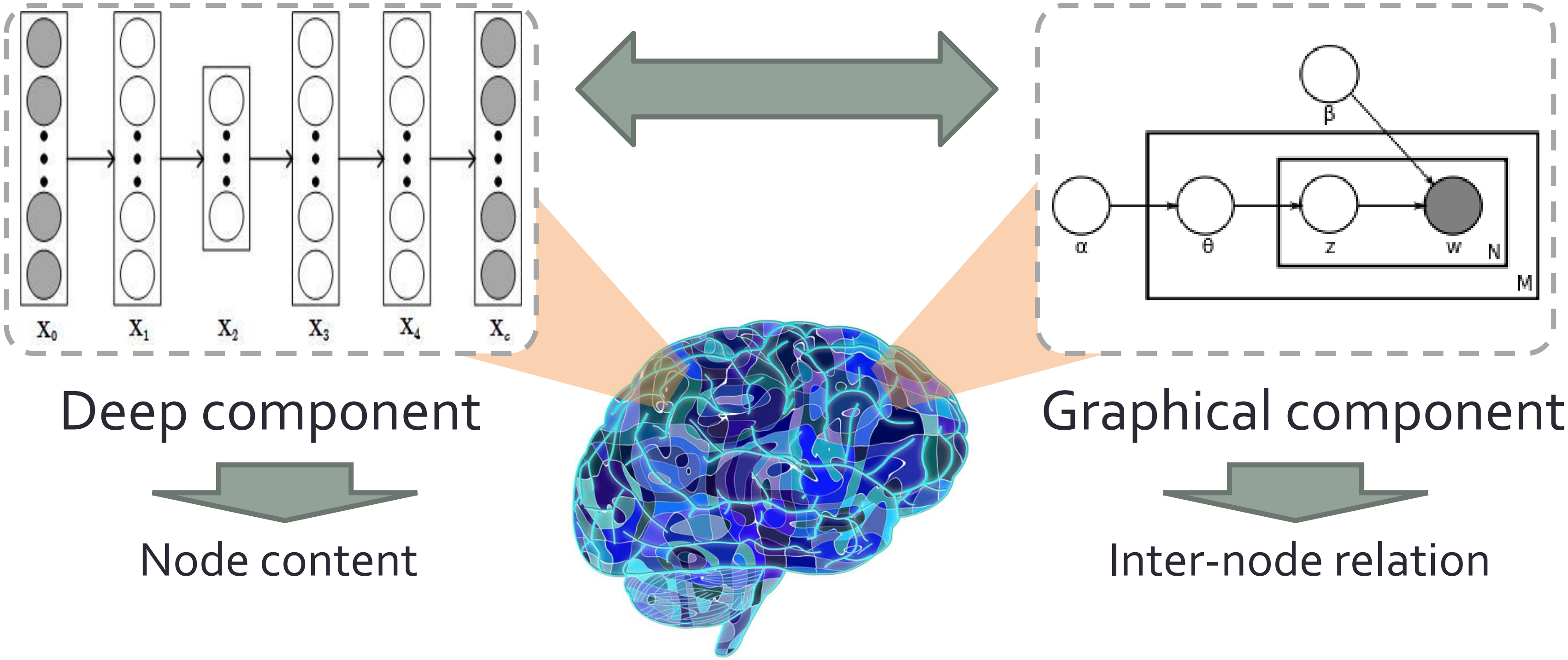
Correctly predict three more tags

Example Movie	Title: E.T. the Extra-Terrestrial	
	RSDAE (ours)	True?
Top 10 tags	**1. Steven Spielberg**	yes
	2. Saturn Award (Best Special Effects)	yes
	3. Saturn Award (Best Writing)	yes
	4. Oscar (Best Editing)	no
	5. Want	no
	6. Liam Neeson	no
	7. AFI 100 (Cheers)	yes
	8. Oscar (Best Sound)	yes
	9. Saturn Award (Best Director)	no
	10. Oscar (Best Music - Original Score)	yes

Very difficult to discover this tag

Does not appear in any related movies

Summary: Relational Autoencoder



Deep component

Node content

Graphical component

Inter-node relation

BDL-Based Relational Autoencoder

Unified into a probabilistic relational model for relational deep learning

[**Wang** et al. AAAI 2015]

Contribution of Relational Probabilistic Autoencoder

1. First deep learning model in the relational domain (graphs)

2. Naturally handle multiple graphs

3. Application to article tagging demonstrating better performance

[**Wang** et al. AAAI 2015]

References

- Bayesian probabilistic matrix factorization using Markov chain Monte Carlo. Salakhutdinov, Ruslan, and Mnih, Andriy. The 25th International Conference on Machine Learning (**ICML**) 2008.
- Probabilistic Matrix Factorization. Salakhutdinov, Ruslan, and Mnih, Andriy. Advances in Neural Information Processing Systems (**NIPS**). 2008.
- Bayesian Learning via Stochastic Gradient Langevin Dynamics. Welling, Max, and Yee W. Teh. The 28th International Conference on Machine Learning (**ICML**). 2011.
- Bayesian Dark Knowledge. Korattikara, Anoop, Vivek Rathod, Kevin Murphy, and Max Welling. Advances in Neural Information Processing Systems (**NIPS**). 2015.

References

- A survey on Bayesian deep learning. Hao Wang, Dit-Yan Yeung. *ACM Computing Surveys* (***CSUR***), 2020.
- Towards Bayesian deep learning: a framework and some existing methods. Hao Wang, Dit-Yan Yeung. *IEEE Transactions on Knowledge and Data Engineering* (***TKDE***), 2016.
- Collaborative deep learning for recommender systems. Hao Wang, Naiyan Wang, Dit-Yan Yeung. *Twenty-First ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (***KDD***), 2015.
- Collaborative recurrent autoencoder: recommend while learning to fill in the blanks. Hao Wang, Xingjian Shi, Dit-Yan Yeung. *Thirtieth Annual Conference on Neural Information Processing Systems* (***NIPS***), 2016.
- Relational stacked denoising autoencoder for tag recommendation. Hao Wang, Xingjian Shi, Dit-Yan Yeung. *Twenty-Ninth AAAI Conference on Artificial Intelligence* (***AAAI***), 2015.
- Relational deep learning: A deep latent variable model for link prediction. Hao Wang, Xingjian Shi, Dit-Yan Yeung. *Thirty-First AAAI Conference on Artificial Intelligence* (***AAAI***), 2017.
- Natural parameter networks: a class of probabilistic neural networks. Hao Wang, Xingjian Shi, Dit-Yan Yeung. *Thirtieth Annual Conference on Neural Information Processing Systems* (***NIPS***), 2016.
- Bidirectional inference networks: A class of deep Bayesian networks for health profiling. Hao Wang, Chengzhi Mao, Hao He, Mingmin Zhao, Tommi S. Jaakkola, Dina Katabi. *Thirty-Third AAAI Conference on Artificial Intelligence* (***AAAI***), 2019.
- Continuously indexed domain adaptation. Hao Wang*, Hao He*, Dina Katabi. *Thirty-Seventh International Conference on Machine Learning* (***ICML***), 2020.
- Assessment of medication self-administration using artificial intelligence. Mingmin Zhao*, Kreshnik Hoti*, Hao Wang, Aniruddh, Raghu, Dina Katabi. ***Nature Medicine***, 2021.