Collaborative Deep Learning For Recommender Systems

Hao Wang, Naiyan Wang, Dit-Yan Yeung
Hong Kong University of Science and Technology

Recommender Systems' Problem

- Observable: A matrix of ratings for user, item pairs.
- This matrix is mostly a sparse matrix and has many missing/unknown values
- Objective: Recommend a user items which are likeable to the user

Recommender Systems

- Content-Based Methods
- Collaborative Filtering
- Hybrid Methods
    - Loosely Coupled
    - Tightly Coupled

Tightly Coupled

- Collaborative Topic Regression (CTR)
  - Probabilistic Matrix Factorization
  - Topic Model: Latent Dirichlet Allocation
- Collaborative Deep Learning (CDL)
  - Probabilistic Matrix Factorization
  - Stacked Denoising Autoencoders

Probabilistic Matrix Factorization

2 Probabilistic Matrix Factorization (PMF)

Suppose we have M movies, N users, and integer rating values from 1 to K[1]. Let R_{ij} represent the rating of user i for movie j, $U \in R^{D\times N}$ and $V \in R^{D\times M}$ be latent user and movie feature matrices, with column vectors U_i and V_j representing user-specific and movie-specific latent feature vectors respectively. Since model performance is measured by computing the root mean squared error (RMSE) on the test set we first adopt a probabilistic linear model with Gaussian observation noise (see fig. 1, left panel). We define the conditional distribution over the observed ratings as

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

[1]Real-valued ratings can be handled just as easily by the models described in this paper.

2

where $\mathcal{N}(x|\mu,\sigma^2)$ is the probability density function of the Gaussian distribution with mean μ and variance σ^2, and I_{ij} is the indicator function that is equal to 1 if user i rated movie j and equal to 0 otherwise. We also place zero-mean spherical Gaussian priors [1, 11] on user and movie feature vectors:

$$p(U|\sigma_U^2) = \prod_{i=1}^{N}\mathcal{N}(U_i|0,\sigma_U^2\mathbf{I}), \qquad p(V|\sigma_V^2) = \prod_{j=1}^{M}\mathcal{N}(V_j|0,\sigma_V^2\mathbf{I}). \tag{2}$$

Denoising Autoencoders

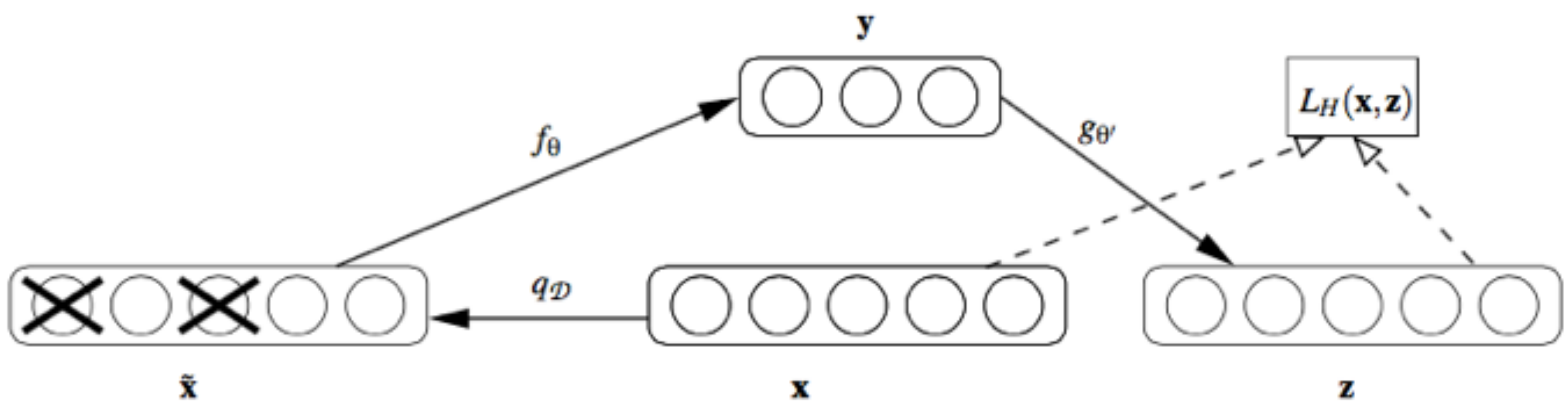


Figure 1: The denoising autoencoder architecture. An example $\mathbf{x}$ is stochastically corrupted (via $q_{\mathcal{D}}$) to $\tilde{\mathbf{x}}$. The autoencoder then maps it to $\mathbf{y}$ (via encoder f_θ) and attempts to reconstruct $\mathbf{x}$ via decoder $g_{\theta'}$, producing reconstruction $\mathbf{z}$. Reconstruction error is measured by loss $L_H(\mathbf{x},\mathbf{z})$.

Stacked Denoising Autoencoders

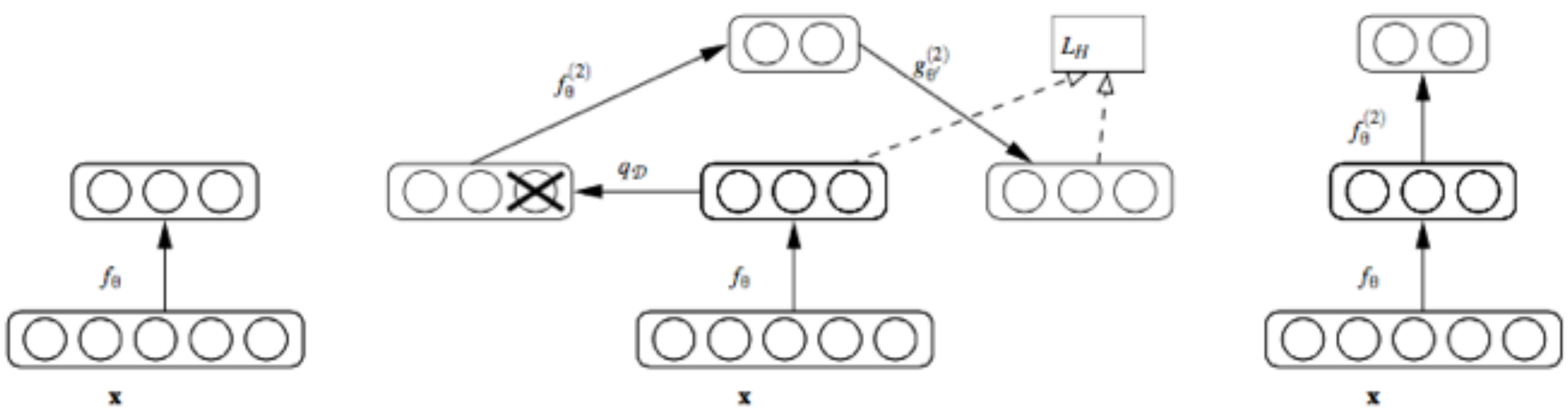
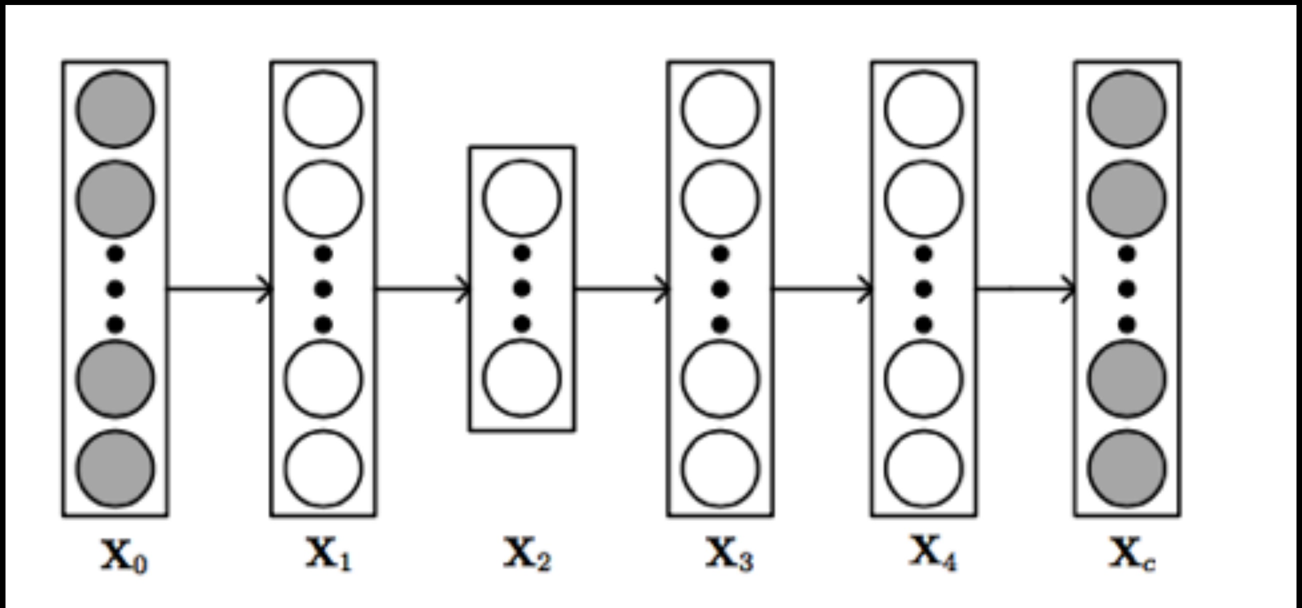


Figure 3: Stacking denoising autoencoders. After training a first level denoising autoencoder (see Figure 1) its learnt encoding function f_θ is used on clean input (left). The resulting representation is used to train a second level denoising autoencoder (middle) to learn a second level encoding function $f_\theta^{(2)}$. From there, the procedure can be repeated (right).

SDAE 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

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

2. For each item j, draw a clean input [1]

   $$\mathbf{X}_{c,j*} \sim \mathcal{N}(\mathbf{X}_{L,j*}, \lambda_n^{-1}\mathbf{I}_J).$$

Collaborative Deep Learning

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:

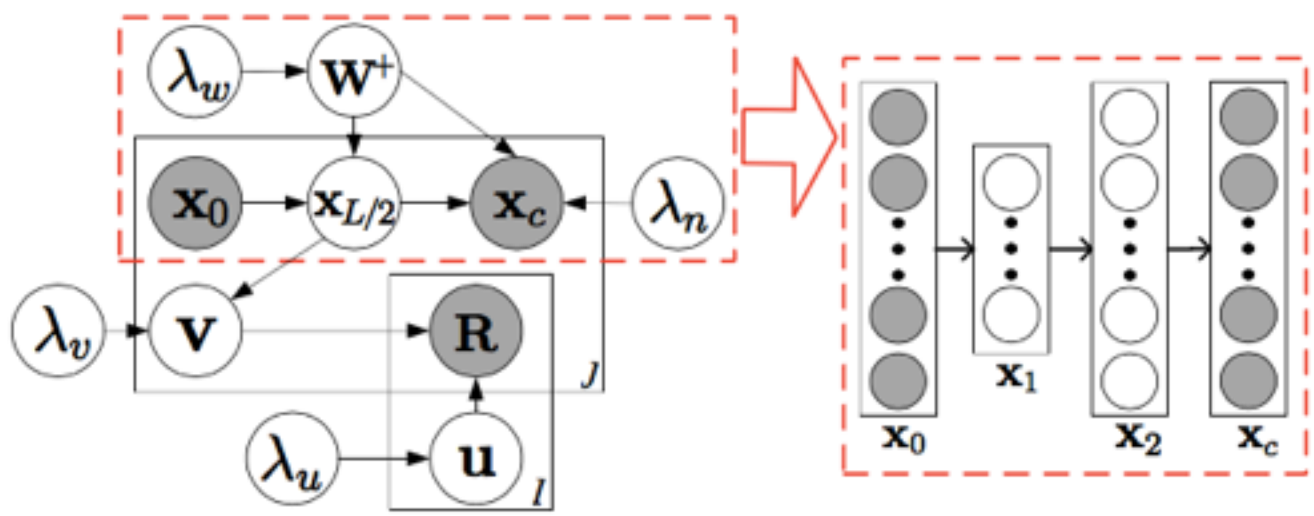
   $$\mathbf{v}_j = \boldsymbol{\epsilon}_j + \mathbf{X}^T_{\frac{L}{2},j*}.$$

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):

   $$\mathbf{R}_{ij} \sim \mathcal{N}(\mathbf{u}_i^T\mathbf{v}_j, \mathbf{C}_{ij}^{-1}).$$



Maximum A Posteriori Estimates

Joint log-likelihood of parameters

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

Maximum A Posteriori Estimates

Learning

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

$$\begin{aligned}
&\nabla_{\mathbf{W}_l}\mathscr{L} = -\lambda_w \mathbf{W}_l \\
&-\lambda_v \sum_j \nabla_{\mathbf{W}_l} f_e(\mathbf{X}_{0,j*}, \mathbf{W}^+)^T (f_e(\mathbf{X}_{0,j*}, \mathbf{W}^+)^T - \mathbf{v}_j) \\
&-\lambda_n \sum_j \nabla_{\mathbf{W}_l} f_r(\mathbf{X}_{0,j*}, \mathbf{W}^+)(f_r(\mathbf{X}_{0,j*}, \mathbf{W}^+) - \mathbf{X}_{c,j*})
\end{aligned}$$

$$\begin{aligned}
&\nabla_{\mathbf{b}_l}\mathscr{L} = -\lambda_w \mathbf{b}_l \\
&-\lambda_v \sum_j \nabla_{\mathbf{b}_l} f_e(\mathbf{X}_{0,j*}, \mathbf{W}^+)^T (f_e(\mathbf{X}_{0,j*}, \mathbf{W}^+)^T - \mathbf{v}_j) \\
&-\lambda_n \sum_j \nabla_{\mathbf{b}_l} f_r(\mathbf{X}_{0,j*}, \mathbf{W}^+)(f_r(\mathbf{X}_{0,j*}, \mathbf{W}^+) - \mathbf{X}_{c,j*}).
\end{aligned}$$

Prediction

$$\mathbf{R}^*_{ij} \approx (\mathbf{u}^*_j)^T (f_e(\mathbf{X}_{0,j*}, \mathbf{W}^{+*})^T + \boldsymbol{\epsilon}^*_j) = (\mathbf{u}^*_i)^T \mathbf{v}^*_j.$$

Some Results

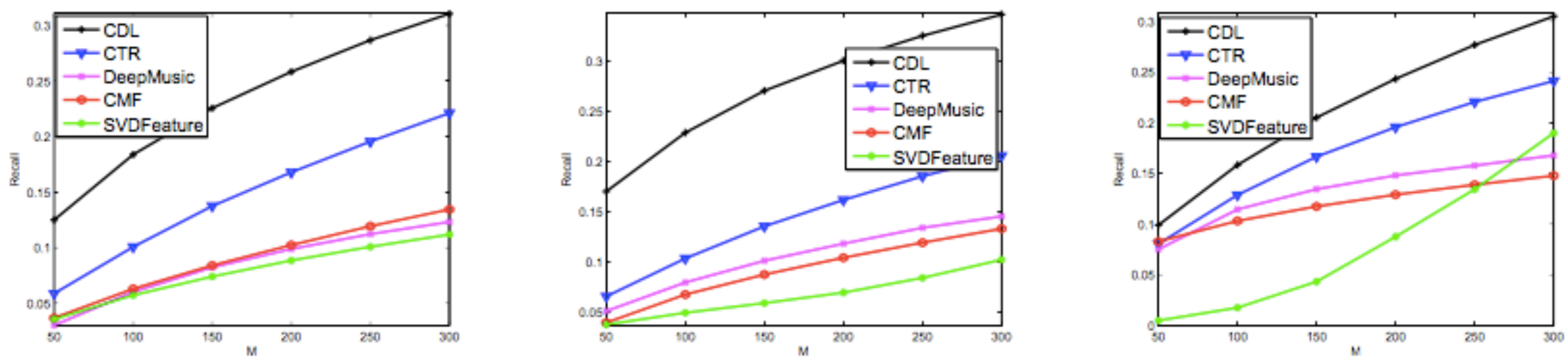


Figure 4: Performance comparison of CDL, CTR, DeepMusic, CMF, and SVDFeature based on recall@M for datasets *citeulike-a*, *citeulike-t*, and *Netflix* in the sparse setting. A 2-layer CDL is used.

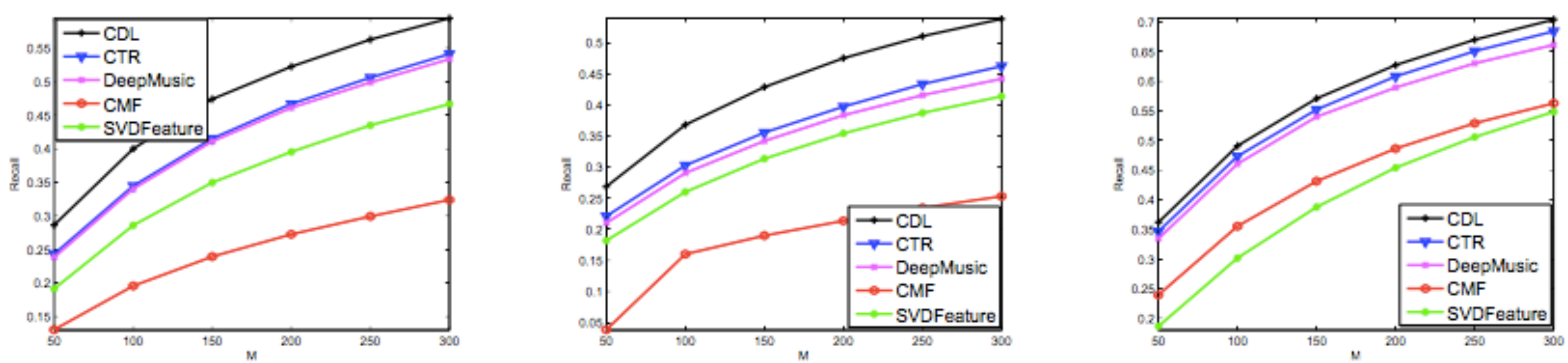


Figure 5: Performance comparison of CDL, CTR, DeepMusic, CMF, and SVDFeature based on recall@M for datasets *citeulike-a*, *citeulike-t*, and *Netflix* in the dense setting. A 2-layer CDL is used.