
# CF-NADE

## A Neural Autoregressive Approach to Collaborative Filtering

*2018. 10. 31.*

*Joonyoung Yi*
*joonyoung.yi@kaist.ac.kr*

**This slide is adopted from the author's slide.*

KAIST

# CONTENTS

# Today's Papers

- NADE (Neural Autoregressive Density Estimator)
  - 2011 AISTAT.
  - Alternative to Restricted Boltzmann Machine (RBM).

- **CF-NADE** (NADE to Collaborative Filtering)
  - 2016 ICML.
  - Apply NADE to Collaborative Filtering (CF) Problem.
  - Matrix Completion is one of the popular algorithms to solve CF Problem.

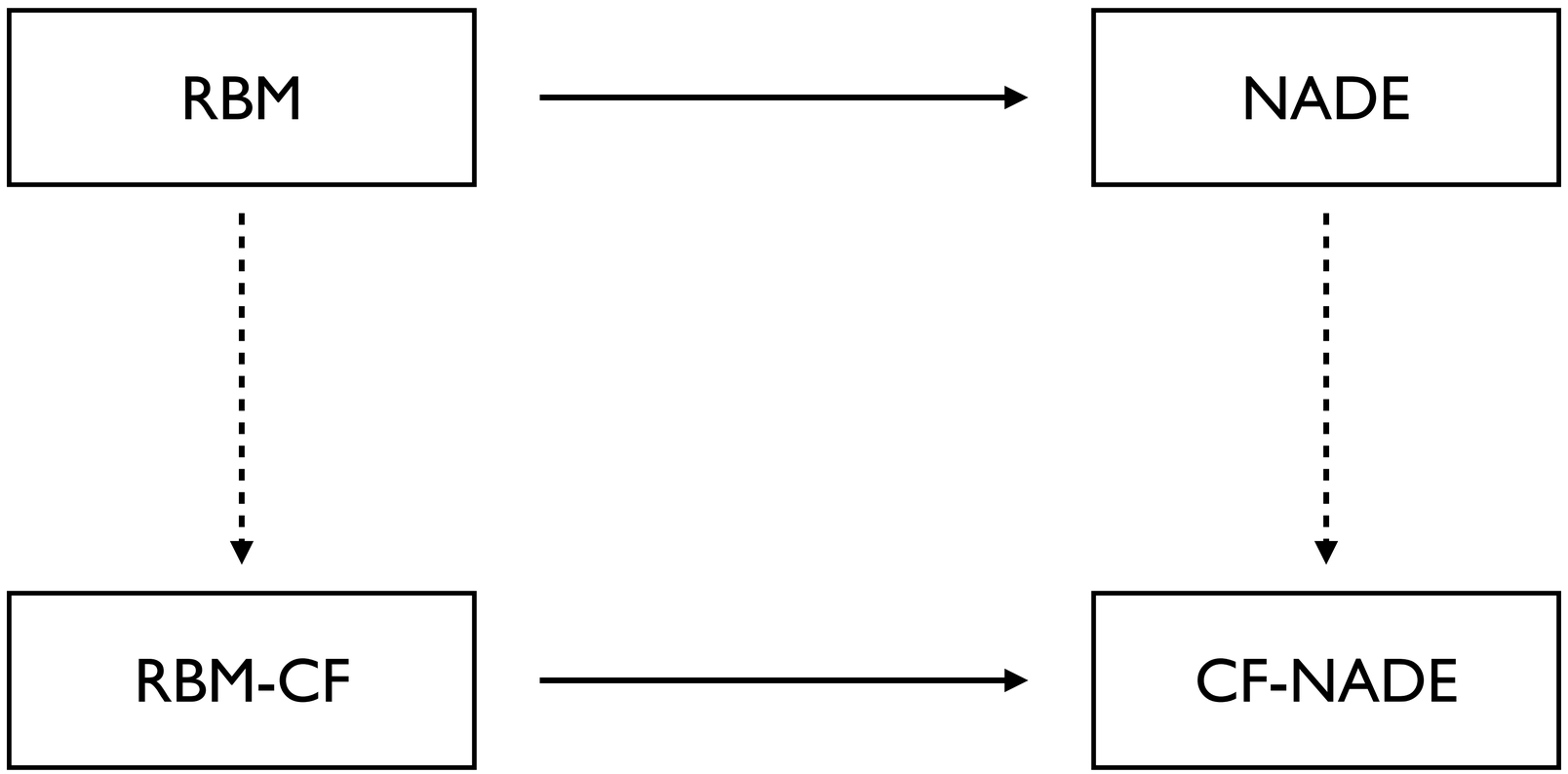
# Motivation

- The Netflix recommendation engine use hybrid models combined with matrix factorization based models (Biased-MF, …) and deep networks [3].
  - Not Netflix Prize.
- The main model used for deep networks is RBM-CF [9].
  - The RBM-CF apply RBM to CF problem.
  - An algorithm that works well with practical data.

- Authors of CF-NADE were in Hulu, which is a competitor of the Netflix.
- To enhance their recommendation engine.
- By replacing RBM-CF to CF-NADE.

- Personally, the CF-NADE paper is able to get several insights on engineering.

# Motivation



# Contents

# RBM TO NADE

- RBM is not tractable by the partition function.

$$E(\mathbf{v}, \mathbf{h}) = -\mathbf{h}^\top \mathbf{W}\mathbf{v} - \mathbf{b}^\top \mathbf{v} - \mathbf{c}^\top \mathbf{h}$$

$$p(\mathbf{v}) = \sum_{\mathbf{h}} \exp(-E(\mathbf{v}, \mathbf{h}))/Z\,,$$

Z is the partition function.
We should calculate Z by considering all possible v, h

- The goal of NADE:
  - Make RBM be tractable.
  - To do this, the author of NADE convert RBM to a Bayesian network.

$$p(\mathbf{v}) = \prod_{i=1}^{D} p(v_i | \mathbf{v}_{\text{parents}(i)})$$

The thing we need to train in NADE

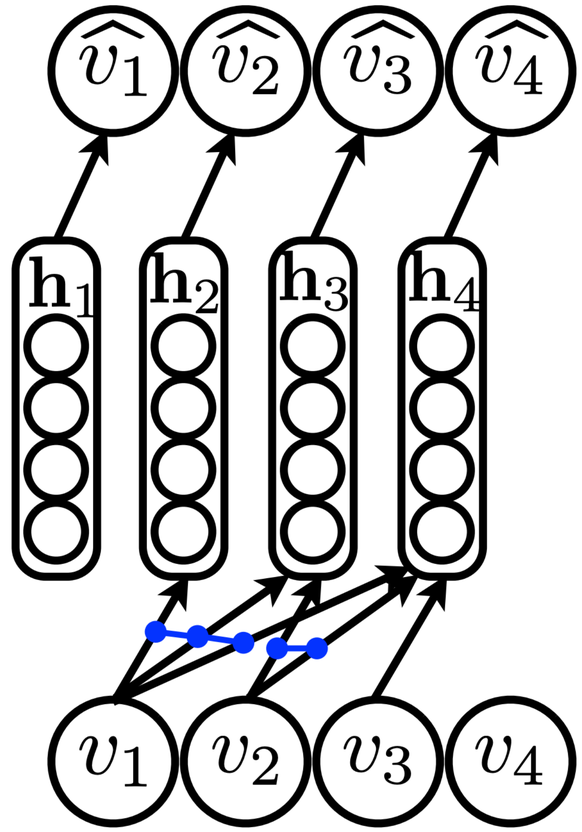
# NADE

- The model of NADE:

$$p(v_i = 1|\mathbf{v}_{<i}) = \text{sigm}\left(b_i + (\mathbf{W}^\top)_{i,.}\mathbf{h}_i\right)$$

$$\mathbf{h}_i = \text{sigm}\left(\mathbf{c} + \mathbf{W}_{.,<i}\mathbf{v}_{<i}\right),$$

$$\frac{1}{T}\sum_{t=1}^{T} -\log p(\mathbf{v}_t) = \frac{1}{T}\sum_{t=1}^{T}\sum_{i=1}^{D} -\log p(v_i|\mathbf{v}_{<i}),$$




  - It is a tractable. There is no partition function or similar things.
  - For fixed length binary vectors.
  - Practical for high dimensional data.

- Practical version of NADE: $p(v_i = 1|\mathbf{v}_{<i}) = \text{sigm}(b_i + \mathbf{V}_{i,:}\mathbf{h}_i)$

$$\mathbf{h}_i = \text{sigm}(\mathbf{c} + \sum_{j<i} W_{:,j})$$

# CONTENTS

# Notation and Probabilistic Model

- The training case for user *u*: $\mathbf{r}^u = (r^u_{m_{o_1}}, r^u_{m_{o_2}}, \ldots, r^u_{m_{o_D}})$
  - *o* is a *D*-tuple in the set of permutations of *(1, 2, …, D)*.
  - The items $m_i \in \{1, 2, \ldots, M\}$.
  - $r^u_{m_{o_i}} \in \{1, 2, \ldots, K\}$.
  - For simplicity, the index *u* of $r^u$ can be omitted.
- We want to model the below equation by NADE.

$$p(\mathbf{r}) = \prod_{i=1}^{D} p\left(r_{m_{o_i}} | \mathbf{r}_{m_{o<i}}\right)$$

  - $\mathbf{r}_{m_{o<i}} = (r_{m_{o_1}}, r_{m_{o_2}}, \ldots, r_{m_{o_{i-1}}})$ : the first *i-1* elements of **r** indexed by *o*.
  - *D* is the number of ratings by user *u*.
  - *o*: ordering.

# The Basic Model

- The objective function:

$$-\log p\left(\mathbf{r}\right) = -\sum_{i=1}^{D} \log p\left(r_{m_{o_i}} | \mathbf{r}_{m_{o<i}}\right)$$

  - where

$$p\left(r_{m_{o_i}} = k | \mathbf{r}_{m_{o<i}}\right) = \frac{\exp\left(s^k_{m_{o_i}}\left(\mathbf{r}_{m_{o<i}}\right)\right)}{\sum_{k'=1}^{K} \exp\left(s^{k'}_{m_{o_i}}\left(\mathbf{r}_{m_{o<i}}\right)\right)}$$

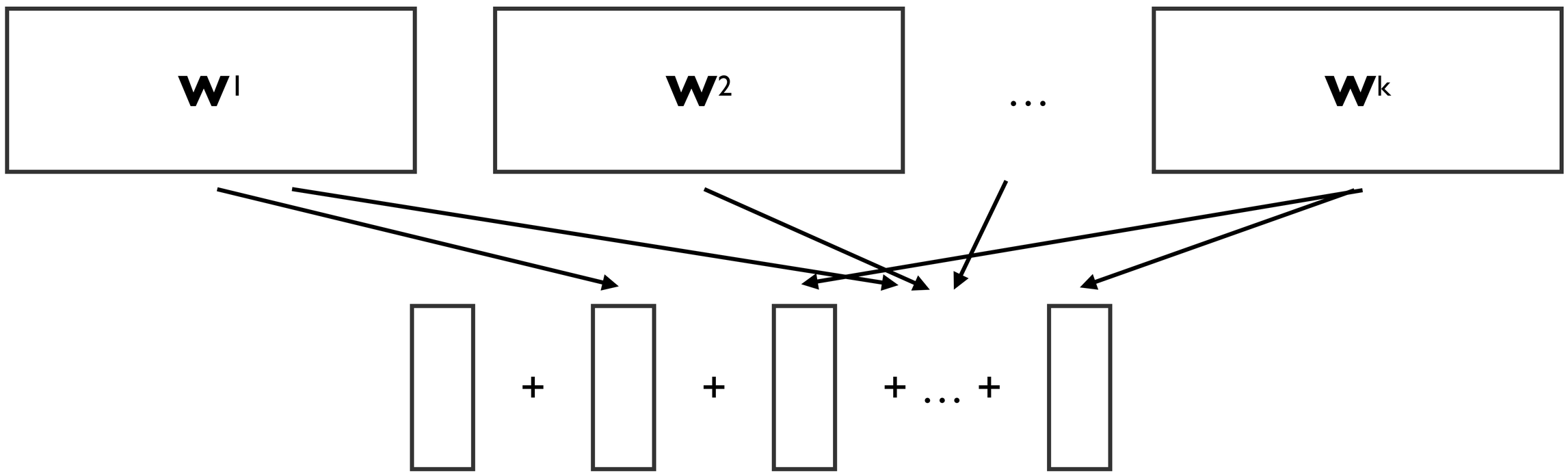
$$s^k_{m_{o_i}}\left(\mathbf{r}_{m_{o<i}}\right) = b^k_{m_{o_i}} + \mathbf{V}^k_{m_{o_i},:}\mathbf{h}\left(\mathbf{r}_{m_{o<i}}\right)$$ no activation function

$$\mathbf{h}\left(\mathbf{r}_{m_{o<i}}\right) = \mathbf{g}\left(\mathbf{c} + \sum_{j<i} \mathbf{W}^{r_{m_{o_j}}}_{:,m_{o_j}}\right)$$

- g( ) : activation function like tanh.
- $\mathbf{W}^k \in \mathbb{R}^{H \times M}$ is a latent matrix. $\mathbf{V}^k \in \mathbb{R}^{M \times H}$ is a weight matrix.
- $\mathbf{b}^k \in \mathbb{R}^M$ and $\mathbf{c} \in \mathbb{R}^H$ are the bias terms.

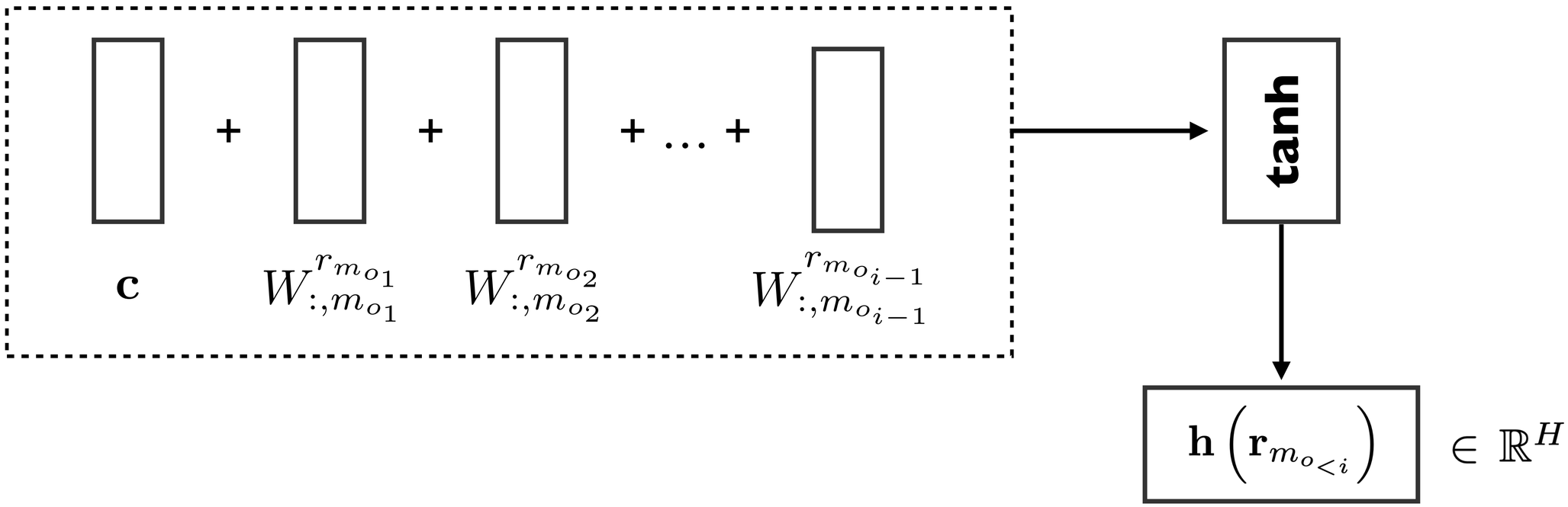
# FIGURES OF THE MODEL

# FIGURES OF THE MODEL

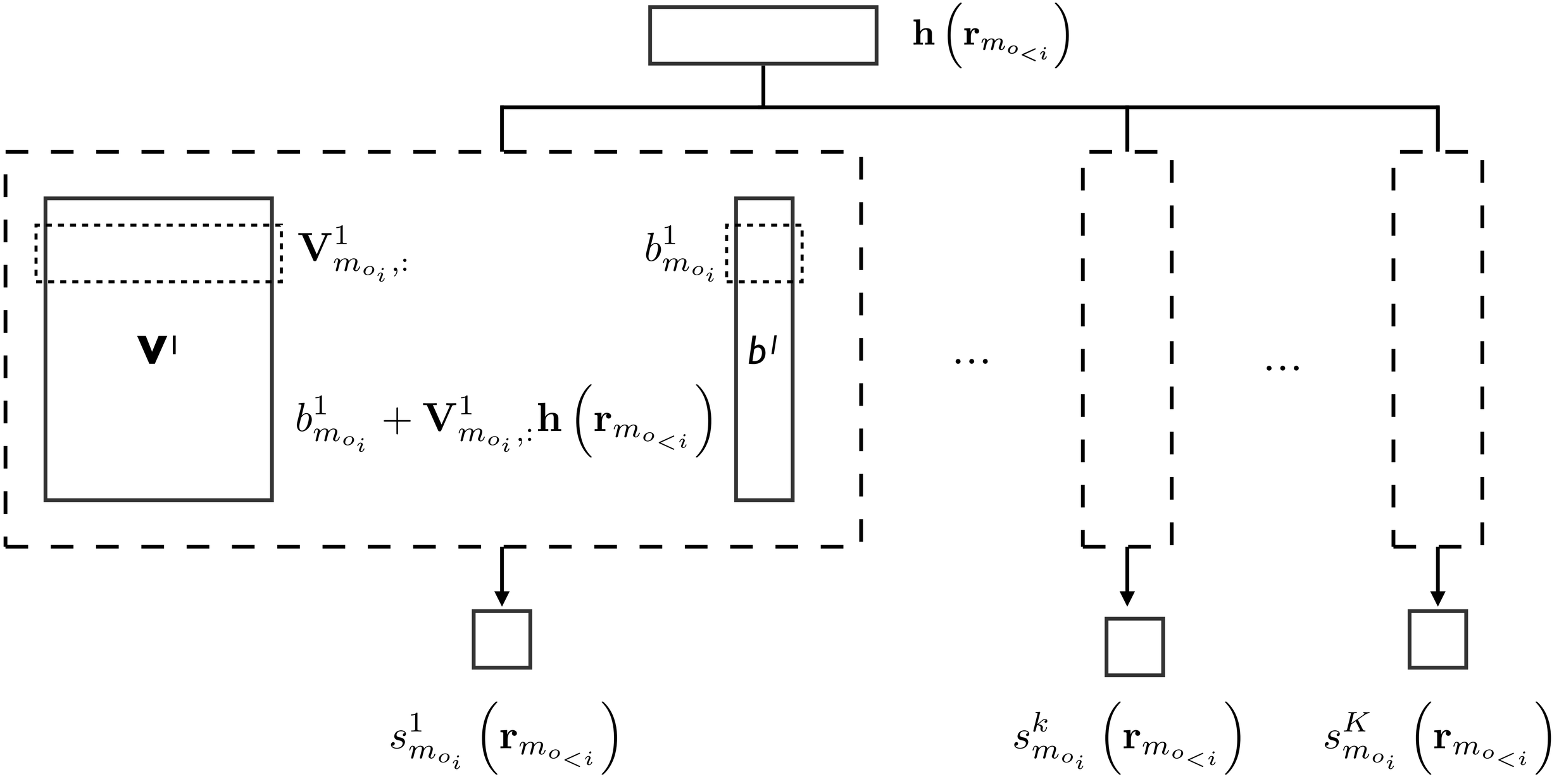



$$\mathbf{c} \in \mathbb{R}^{H} \quad W_{:,m_{o_1}}^{r_{m_{o_1}}} \quad W_{:,m_{o_2}}^{r_{m_{o_2}}} \quad W_{:,m_{o_{i-1}}}^{r_{m_{o_{i-1}}}}$$

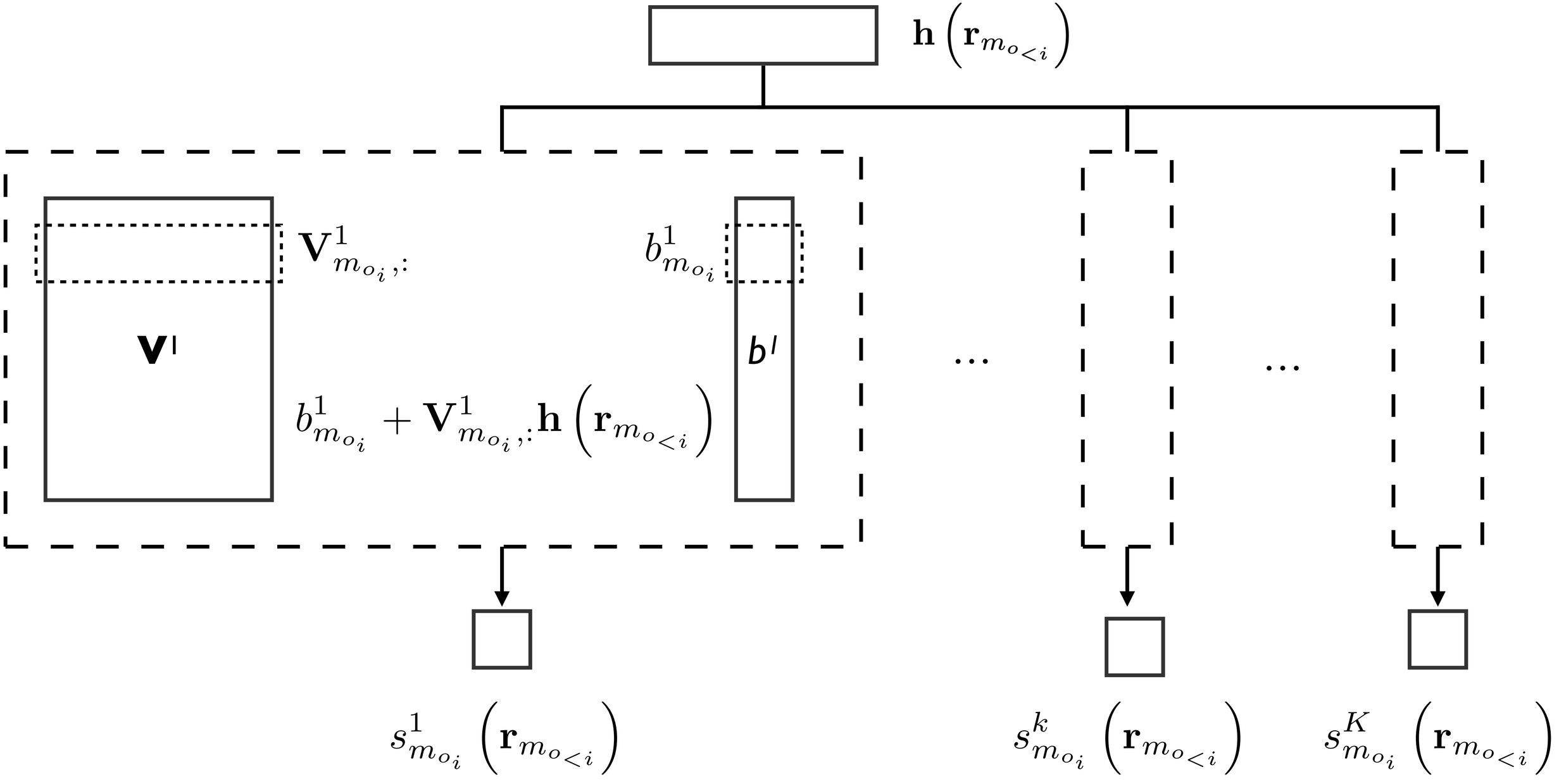
# FIGURES OF THE MODEL



# FIGURES OF THE MODEL



# FIGURES OF THE MODEL




Then, we can calculate $p\left(r_{m_{o_i}} = k|\mathbf{r}_{m_{o<i}}\right)$ by softmax.

It can be viewed as single-hidden-layer MLP.

# The Order of the Ratings

- At fist, the authors used timestamps to make order.
  - The benchmark datasets (e.g,. Movielens, Netflix) contains timestamps information.

- They found that a random order works well in practice.
  - For fair comparison with other algorithms.
  - There was no significant difference from using timestamp information.

- The order has to be prefixed before training.

# Test Phase for Basic Model

- Training phase:

$$-\log p\left(\mathbf{r}\right)=-\sum_{i=1}^{D}\log p\left(r_{m_{o_i}}|\mathbf{r}_{m_{o<i}}\right)$$

- Test phase:
  - Given a user's past behavior **r** = $\left(r_{m_{o_1}},r_{m_{o_2}},\ldots,r_{m_{o_D}}\right)$.
  - The user's rating of new item *m** can be predicted as:

$$\hat{r}_{m^*}=\mathbb{E}_{p(r_{m^*}=k|\mathbf{r})}\left[k\right]$$

- where

$$\begin{aligned}s_{m^*}^{k}\left(\mathbf{r}\right) &= b_{m^*}^{k}+\mathbf{V}_{m^*,:}^{k}\mathbf{h}\left(\mathbf{r}\right)\\ \mathbf{h}\left(\mathbf{r}\right) &= \mathbf{g}\left(\mathbf{c}+\sum_{j=1}^{D}\mathbf{W}_{:,m_{o_j}}^{r_{m_{o_j}}}\right).\end{aligned}$$

# CONTENTS

# Weight Sharing

- 2 kinds of weight sharing in this paper.
  - Between users
  - Between ratings

- Between users:
  - The CF (Collaborative Filtering) has sparsity problem.
  - The training data of CF problem is too sparse.
  - **W**, **K**, **c**, **b** are shared between users.

- Between ratings:
  - Share weighs between different ratings. **Why? and How?**

# Weight Sharing Between Ratings

- Why?
  - Some items may not be rated much.
  - Items that are less rated are less optimized.
  - To enhance optimization of items that are less rated

- How? using other weights of other ratings.

$$\mathbf{h}\left(\mathbf{r}_{m_{o<i}}\right) = \mathbf{g}\left(\mathbf{c} + \sum_{j<i} \mathbf{W}_{:,m_{o_j}}^{r_{m_{o_j}}}\right) \longrightarrow \mathbf{h}\left(\mathbf{r}_{m_{o<i}}\right) = \mathbf{g}\left(\mathbf{c} + \sum_{j<i} \sum_{k=1}^{r_{m_{o_j}}} \mathbf{W}_{:,m_{o_j}}^{k}\right)$$

$$s_{m_{o_i}}^{k}\left(\mathbf{r}_{m_{o<i}}\right) = b_{m_{o_i}}^{k} + \mathbf{V}_{m_{o_i},:}^{k}\mathbf{h}\left(\mathbf{r}_{m_{o<i}}\right) \longrightarrow s_{m_{o_i}}^{k}\left(\mathbf{r}_{m_{o<i}}\right) = \sum_{j \le k}\left(b_{m_{o_i}}^{j} + \mathbf{V}_{m_{o_i},:}^{j}\mathbf{h}\left(\mathbf{r}_{m_{o<i}}\right)\right)$$

[ Basic model ] [ Weight sharing between ratings ]

# WHY WEIGHT SHARING BETWEEN RATINGS WORKS?

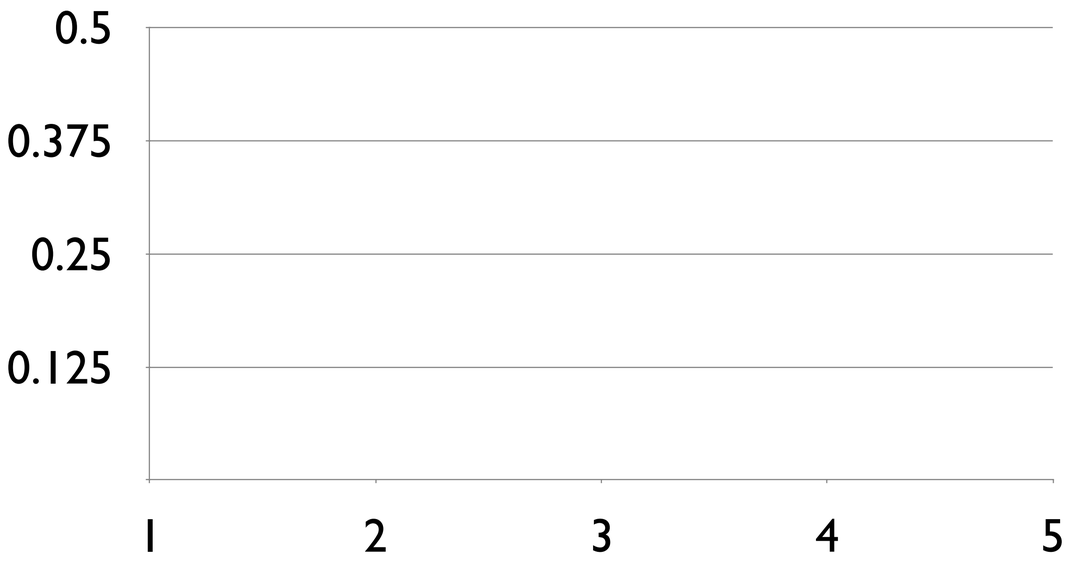
- When calculating the parameter of rating k in weight sharing,
  the parameters of rating 1, ... , k of the basic model are added.

- It is a kind of regularization,
  which encourages the model to use as many parameters as possible to explain the data.

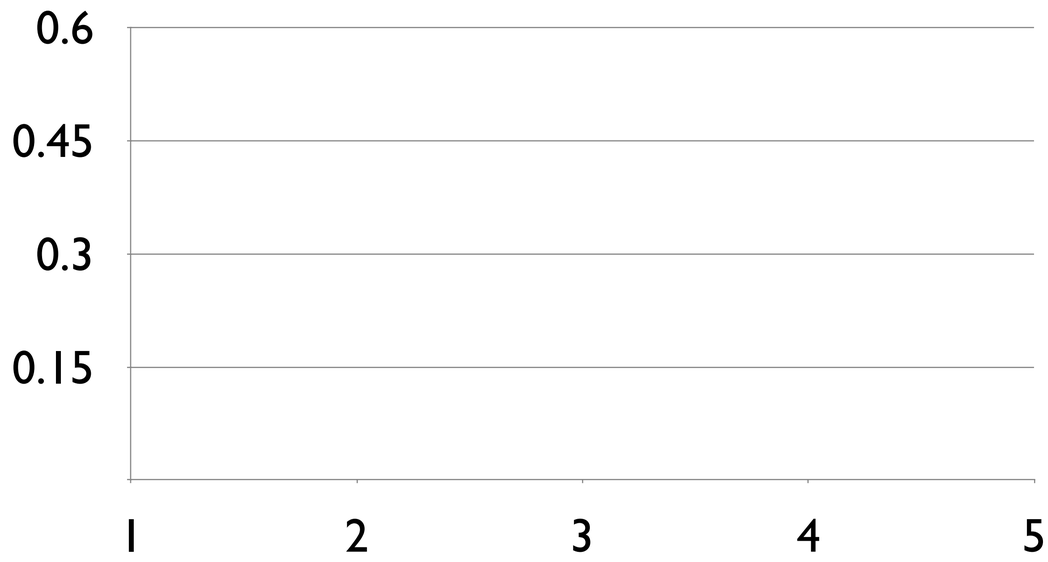
# Contents

# Regular Cost $C_{REG}$ and Ordinal Cost $C_{ORD}$

- In the basic model, we use softmax function to calculate cost.
  - We call it as regular cost $C_{reg}$.
- However, rating data has ordinal nature.
  - If user *u* want to give the movie *m* 4-star score,
    the PMF by ordinal cost is more reasonable compared to that by regular cost.
  - Ordinal cost could be calculated by ranking loss.




[ PMF by Regular Cost $C_{reg}$ ]




[ PMF by Ordinal Cost $C_{ord}$ ]

# Ordinal Cost $C_{ORD}$

- What is ordinal nature?
  - Suppose $r_{m_{o_i}} = k$.
  - the ranking of preferences over all the possible ratings can be expressed as:

$$k \succ k-1 \succ \quad \ldots \quad \succ 1$$
$$k \succ k+1 \succ \quad \ldots \quad \succ K$$

  - $k \succ k-1$ denotes the preference of rating *k* over *k-1*.
    - It is used by multiplying two ranking losses (kind of heuristic I think).
- To capture this ordinal nature:

$$p\left(r_{m_{o_i}} = k | \mathbf{r}_{m_{o<i}}\right) = \frac{\exp\left(s^k_{m_{o_i}}\left(\mathbf{r}_{m_{o<i}}\right)\right)}{\sum_{k'=1}^{K}\exp\left(s^{k'}_{m_{o_i}}\left(\mathbf{r}_{m_{o<i}}\right)\right)}$$

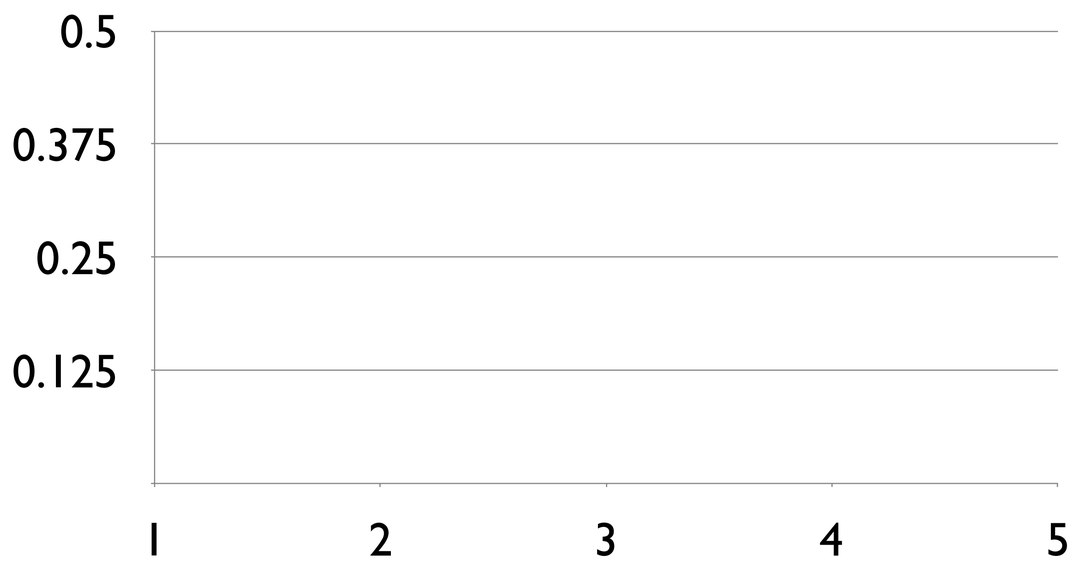
regular to ordinal

$$p\left(r_{m_{o_i}} = k | \mathbf{r}_{m_{o<i}}\right) = \prod_{j=k}^{1}\frac{\exp(s^j_{m_{o_i}})}{\sum_{t=1}^{j}\exp(s^t_{m_{o_i}})}\prod_{j=k}^{K}\frac{\exp(s^j_{m_{o_i}})}{\sum_{t=j}^{K}\exp(s^t_{m_{o_i}})}$$

# Some Notes for Ordinal Cost $C_{ORD}$

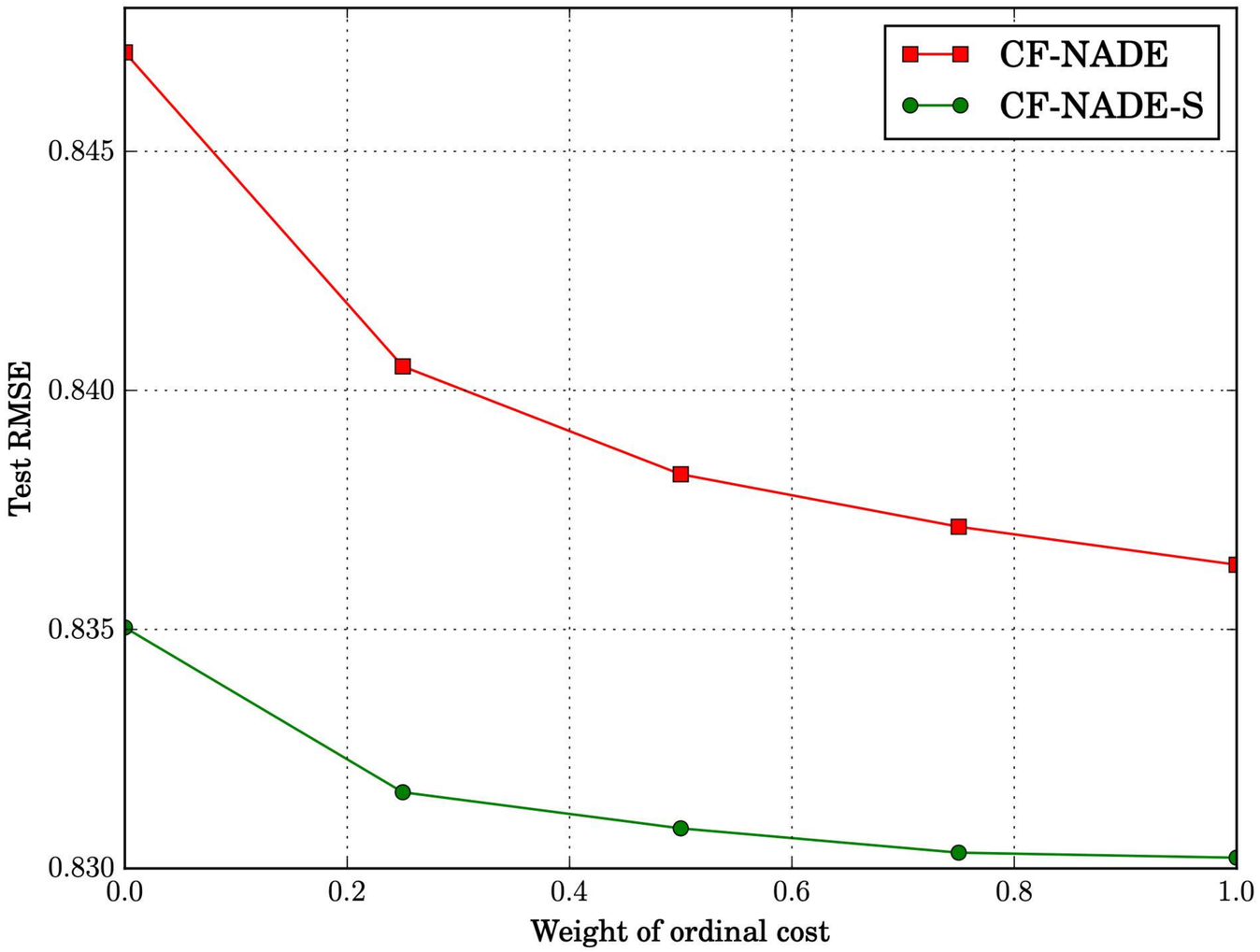
$$p\left(r_{m_{o_i}} = k | \mathbf{r}_{m_{o<i}}\right) = \prod_{j=k}^{1} \frac{\exp(s^j_{m_{o_i}})}{\sum_{t=1}^{j} \exp(s^t_{m_{o_i}})} \prod_{j=k}^{K} \frac{\exp(s^j_{m_{o_i}})}{\sum_{t=j}^{K} \exp(s^t_{m_{o_i}})}$$

- 1. Ordinal cost is inspired by Xia et al (2008) [4].
- 2. the equation of PMF by ordinal cost is not PMF.
  - We have to normalize!
- 3. the ordinal cost can not always capture ordinal nature.
  - ex. $s^1, s^2, s^3, s^4, s^5 = 1, 2.7, 2, 3, 1$



# The Effects of Ordinal Cost

- Let hybrid cost: $\mathcal{C}_{\text{hybrid}} = (1-\lambda)\mathcal{C}_{\text{reg}} + \lambda\mathcal{C}_{\text{ord}}$
- CF-NADE: the basic model.
- CF-NADE-S: the basic model with shared parameters.




- The model with weight sharing outperforms the basic model.
- The ordinal cost outperforms the regular cost.

# CONTENTS

# MAKE CF-NADE BE DEEP

- The basic CF-NADE model can be viewed as single-hidden-layer MLP.
- To make CF-NADE model be deep,
  all we need is to define the relation between hidden layers.

$$\mathbf{h}^{(l)}\left(\mathbf{r}_{m_{o<i}}\right) = \mathbf{g}\left(\mathbf{c}^{(l)} + \mathbf{W}^{(l)}\mathbf{h}^{(l-1)}\left(\mathbf{r}_{m_{o<i}}\right)\right)$$

  - This architecture is adopted from Uria et al. (2014), which makes original NADE be deep networks [5].

The basic model of CF-NADE

$$s^k_{m_{o_i}}\left(\mathbf{r}_{m_{o<i}}\right) = b^k_{m_{o_i}} + \mathbf{V}^k_{m_{o_i},:}\mathbf{h}\left(\mathbf{r}_{m_{o<i}}\right)$$

$$\mathbf{h}\left(\mathbf{r}_{m_{o<i}}\right) = \mathbf{g}\left(\mathbf{c} + \sum_{j<i}\mathbf{W}^{r_{m_{o_j}}}_{:,m_{o_j}}\right)$$

# To Train Deep Networks

- Training deep networks is difficult compared to training the basic model.
  - It has the quite large numbers of free parameters.

- This paper suggest 2 tricks to handle the situation training deep networks.
  - 1. Data augmentation
  - 2. Reduction of free parameters in the networks.

# Data Augmentation

- As I mentioned earlier, a random order works well in practice.
- Different order is an different instantiation of CF-NADE for the same user.
  - This is key to extend CF-NADE to a deep model.

- Training all possible orderings:

$$\mathcal{C} = \mathbb{E}_{o \in \mathcal{O}} \sum_{i=1}^{D} -\log p\left(\mathbf{r}_{m_{o_i}} | \mathbf{r}_{m_{o_{<i}}}, o\right)$$

  - Given a context $\mathbf{r}_{m_{o_{<i}}}$, CF-NADE be equally good at modeling $\mathbf{r}_{m_{o_i}}$.

- Cost function is rewritten as:

$$\mathcal{C} = \frac{D}{D-i+1} \sum_{j \geq i} -\log p\left(r_{m_{o_j}} | \mathbf{r}_{m_{o_{<i}}}\right)$$

# Reduction of Free Parameters

- Too many parameters…
  - $\mathbf{W}^k \in \mathbb{R}^{H \times M} \qquad \mathbf{V}^k \in \mathbb{R}^{M \times H} \qquad k \in \{1, 2, ..., K\}$
  - Netflix dataset, *M=17700, H=500, K=5,* then *89* million free parameters.
  - Weight decay and dropout is ok, but…

- Solution: Factorize *W*, *V* by a product of 2 low-rank matrices.

$$W^k_{i,m} = \sum_{j=1}^{J} B_{i,j} A^k_{j,m} \qquad V^k_{m,i} = \sum_{j=1}^{J} P^k_{m,j} Q_{j,i} \qquad J \ll H \quad J \ll M$$

  - e.g. *J=50, M=17700, H=500, K=5*, then only *9*-million free parameters for Netflix dataset.

- Is this the best way? I think there is better way to make scalability.

# CONTENTS

# EXPERIMENTS

- Metric: $RMSE = \sqrt{\frac{\sum_{i=1}^{S}(r_i - \tilde{r}_i)^2}{S}}$
  - $r_i$ is the i-th true rating and $\tilde{r}_i$ is the predicted rating by the model.
  - S is the total number of ratings in the test set.
  - 90% training set and 10% test set.
- Benchmark dataset:

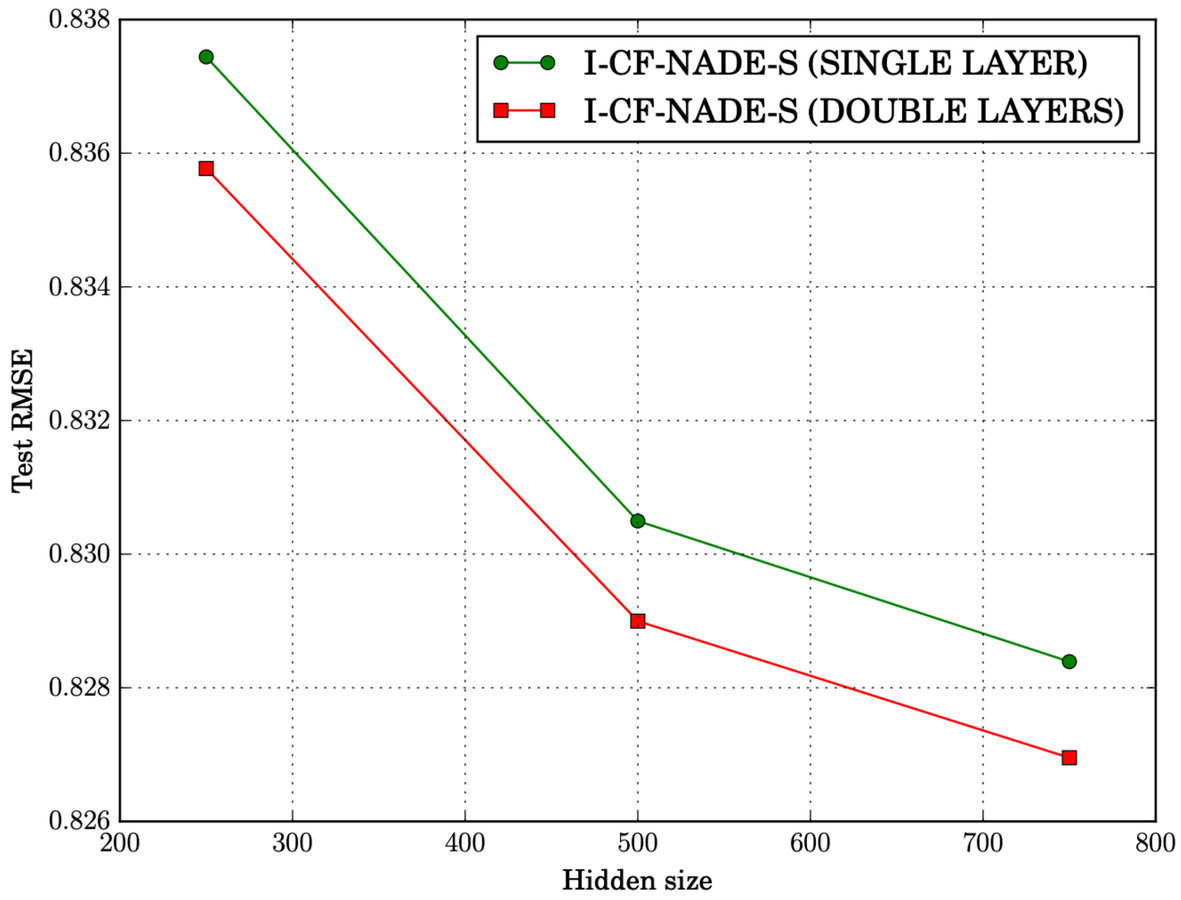
| dataset | #Users | #Items | #Scales | #Ratings |
|---|---|---|---|---|
| Movielens 100k | 1000 | 1700 | 5 | $10^5$ |
| Movielens 1M | 6040 | 3952 | 5 | $10^6$ |
| Movielens 10M | 71567 | 10681 | 10 | $10^7$ |
| Netflix | 480189 | 17770 | 5 | $10^8$ |

# User-Based vs. Item-Based

- The model described in previous slides are user-based model.
- We model rating vectors of user: $\mathbf{r}^u$.

- Similary, item-based CF-NADE model rating vectors of item: $\mathbf{r}^i$.
- Some CF papers said item-based CF model outperforms user-based CF.

# EXPERIMENT RESULTS - MOVIELENS 1M




| METHOD | TEST RMSE |
|---|---|
| PMF† | 0.883 |
| U-RBM∗ | 0.881 |
| U-AUTOREC (SEDHAIN ET AL., 2015) | 0.874 |
| LLORMA-GLOBAL (LEE ET AL., 2013) | 0.865 |
| I-RBM∗ | 0.854 |
| BIASMF∗ | 0.845 |
| NNMF (DZIUGAITE & ROY, 2015) | 0.843 |
| LLORMA-LOCAL (LEE ET AL., 2013) | 0.833 |
| I-AUTOREC (SEDHAIN ET AL., 2015) | 0.831 |
| U-CF-NADE-S (SINGLE LAYER) | 0.850 |
| U-CF-NADE-S (2 LAYERS ) | 0.845 |
| I-CF-NADE-S (SINGLE LAYER) | 0.830 |
| I-CF-NADE-S (2 LAYERS) | **0.829** |

†: Taken from (Dziugaite & Roy, 2015).
∗: Taken from (Sedhain et al., 2015).

- In Movielens 1M dataset, CF-NADE outperforms all state-of-arts algorithms.
- The double hidden layer model outperforms single hidden layer model.
- Item based CF model outperforms user based CF model.

# Experiment Results - Other Datasets

*Table 2.* Test RMSE of different models on MovieLens 10M.

| Method | Test RMSE |
|---|---|
| U-AutoRec (Sedhain et al., 2015) | 0.867 |
| I-RBM† | 0.825 |
| U-RBM† | 0.823 |
| LLORMA-Global (Lee et al., 2013) | 0.822 |
| BiasMF† | 0.803 |
| LLORMA-Local (Lee et al., 2013) | 0.782 |
| I-AutoRec (Sedhain et al., 2015) | 0.782 |
| U-CF-NADE-S (single layer) | 0.772 |
| U-CF-NADE-S (2 layers) | **0.771** |

†: Taken from (Sedhain et al., 2015).

*Table 3.* Test RMSE of different models on Netflix dataset.

| Methods | Test RMSE |
|---|---|
| LLORMA-Global (Lee et al., 2013) | 0.874 |
| U-RBM† | 0.845 |
| BiasMF† | 0.844 |
| LLORMA-Local (Lee et al., 2013) | 0.834 |
| I-AutoRec (Sedhain et al., 2015) | 0.823 |
| U-CF-NADE-S (single layer) | 0.804 |
| U-CF-NADE-S (2 layers) | **0.803** |

†: Taken from (Sedhain et al., 2015).

- CF-NADE shows the best performance even on the larger data sets.
- Now, it is one of the best performing algorithms except MRMA (NIPS 2017).
  - MRMA model is not deep networks, but a matrix factorization based model.
- They didn't use item based CF-NADE because of the computational issue.
  - I think that the performance will be better if we use item-based CF-NADE model.

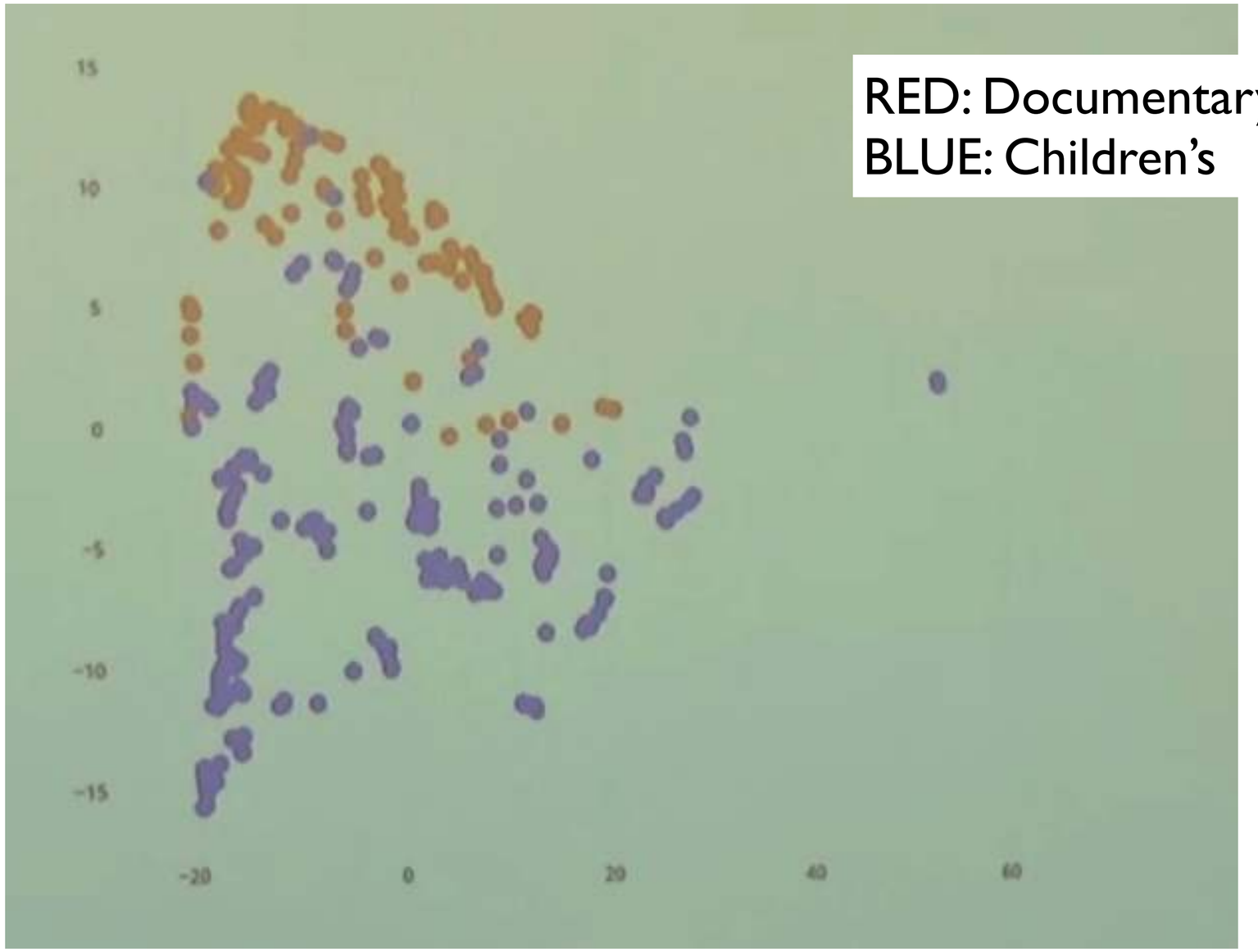
# ITEM-BASED VS USER-BASED

- At first, the RBM-CF paper [9] said that item based CF model outperforms user based CF model.
- The AAE paper [8], AutoRec [7] and CF-NADE [2] paper presented the experimental results supporting the claim.
- There are pros and cons: performance vs computational cost.

- Why item based model outperforms user based model?
  - The AutoRec [7] paper said that
  - The average number of ratings per item is much more than those per user
    - It is inconsistent with the experimental results with movie lens 100k data of other papers.
  - High variance in the number of user ratings leads to less reliable prediction for user-based methods.

# Visualization of the Learned Model

- tSNE of the representations of the movies on Movielens 1M dataset.



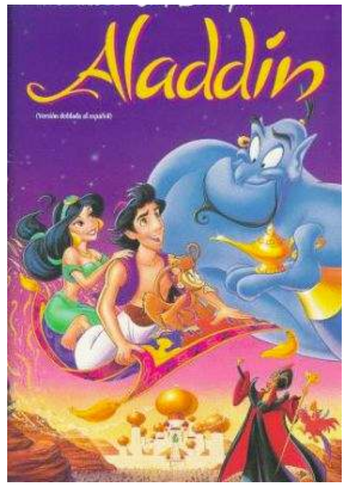
# COSINE SIMILARITIES OF THE *W* VECTOR

- They picked 5 most similar movies of the left side movies

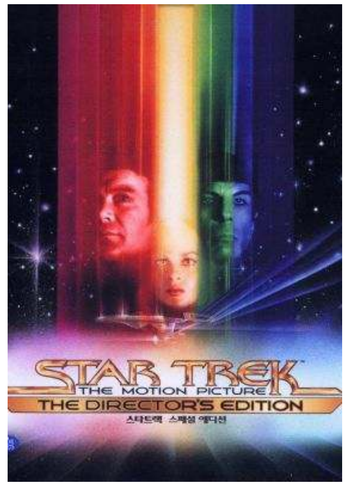

Star Trek VI:
The Undiscovered Country


The Lion King


Star Trek III:
The Search for Spock

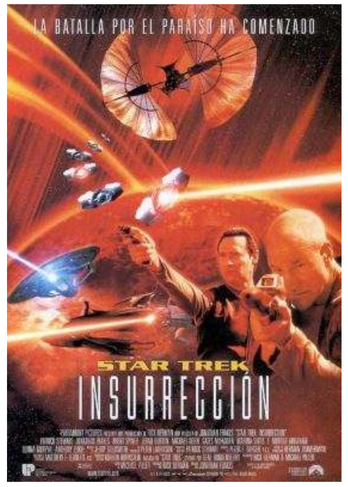



Aladdin

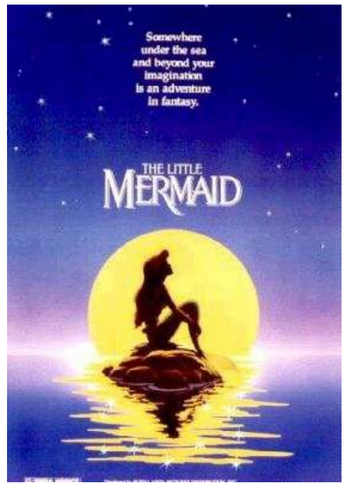



Star Trek:
Generations

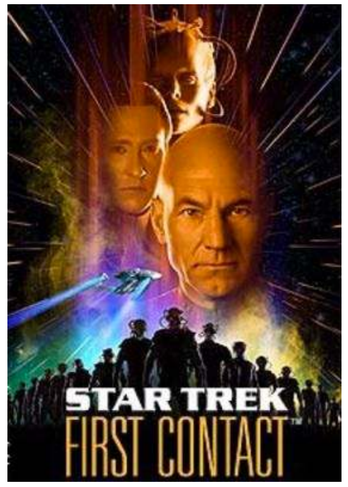

Beauty and
the Beast




Star Trek:
Insurrection

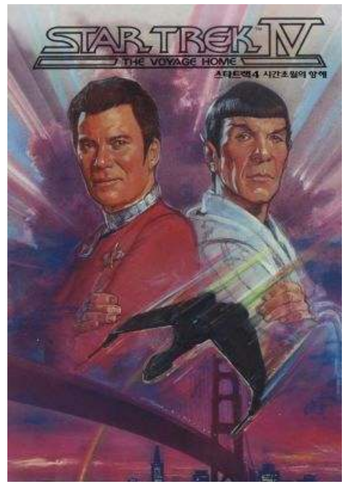



The Little
Mermaid

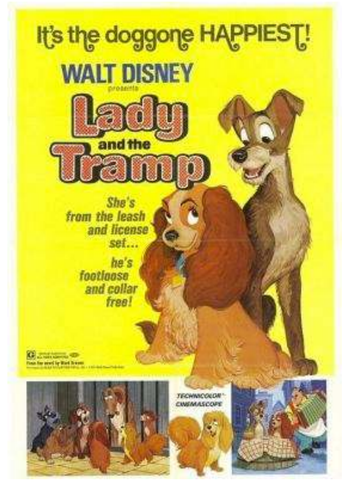



Star Trek:
First Contact


A Bug's Life




Star Trek IV:
The Voyage Home




Lady and the Tramp

# Contents

# Conclusion

- CF-NADE:
  - An efficient and powerful architecture for CF tasks.
  - Sharing parameters is a good way to improve performance.
  - Better scalability by factorization.
  - Can be extended to a deep version and performs well.
- Ordinal cost for rating data
- Meaningful representations
- Source code at: https://github.com/lan09/CF-NADE

# A Follow-up Study

- Implicit Feedback vs Explicit Feedback
  - The recommendation using the implicit feedback data (e.g., click or view).
  - Generally, implicit feedback problem is more difficult than explicit feedback problem.
  - Because, the view or click can not tell the positive preference.

- Implicit CF-NADE [6]:

$$p\left(\mathbf{t}|\mathbf{c}\right) = \prod_{i=1}^{M} p\left(t_i|\mathbf{t}_{<i}, \mathbf{c}_{<i}\right)$$

  - where $\mathbf{t}$ is implicit feedback data and $\mathbf{c}$ is confidence level.

$$t_i^u = \begin{cases} 1 & r_i^u > 0 \\ 0 & r_i^u = 0 \end{cases} \qquad c_i^u = 1 + \alpha r_i^u \qquad \mathcal{C} = -\sum_{i=1}^{M} c_i \log p\left(t_i|\mathbf{t}_{<i}, \mathbf{c}_{<i}\right)$$

# REFERENCE


- [1] Larochelle, Hugo and Murray, Iain. The neural autoregressive distribution estimator. In International Conference on Artificial Intelligence and Statistics, pp. 29–37, 2011.
- [2] Zheng, Y., Tang, B., Ding, W., & Zhou, H. (2016). A neural autoregressive approach to collaborative filtering. *arXiv preprint arXiv:1605.09477*.
- [3] https://www.youtube.com/watch?v=JBP1xUIlH5I&t=4s
- [4] Xia, Fen, Liu, Tie-Yan, Wang, Jue, Zhang, Wensheng, and Li, Hang. Listwise approach to learning to rank: theory and algorithm. In Proceedings of the 25th international conference on Machine learning, pp. 1192–1199. ACM, 2008
- [5] Uria, Benigno, Murray, Iain, and Larochelle, Hugo. A deep and tractable density estimator. JMLR: W&CP, 32(1): 467–475, 2014.
- [6] Zheng, Y., Liu, C., Tang, B., & Zhou, H. (2016, September). Neural autoregressive collaborative filtering for implicit feedback. In *Proceedings of the 1st workshop on deep learning for recommender systems* (pp. 2-6). ACM.
- [7] Sedhain, S., Menon, A. K., Sanner, S., & Xie, L. (2015, May). Autorec: Autoencoders meet collaborative filtering. In *Proceedings of the 24th International Conference on World Wide Web* (pp. 111-112). ACM.
- [8] Lee, K., Lee, Y. H., & Suh, C. (2018, June). Alternating autoencoders for matrix completion. In *2018 IEEE Data Science Workshop (DSW)* (pp. 130-134). IEEE.
- [9] Salakhutdinov, R., Mnih, A., & Hinton, G. (2007, June). Restricted Boltzmann machines for collaborative filtering. In *Proceedings of the 24th international conference on Machine learning* (pp. 791-798). ACM.