IRGAN: A Minimax Game for Unifying Generative and Discriminative Information Retrieval Models

Jun Wang
University College London
j.wang@cs.ucl.ac.uk

Lantao Yu, Weinan Zhang*
Shanghai Jiao Tong University
wnzhang@sjtu.edu.cn

Yu Gong, Yinghui Xu
Alibaba Group
renji.xyh@taobao.com

Benyou Wang, Peng Zhang
Tianjin University
pzhang@tju.edu.cn

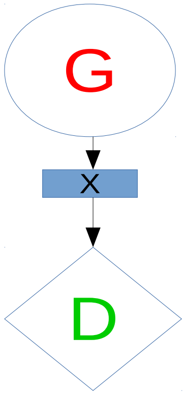
Dell Zhang
Birkbeck, University of London
dell.z@ieee.org

2017 SIGIR

王润生
May 5, 2019

The Idea of GANs

- There is a generator G that captures the data distribution and tries to generate high quality data to fool the discriminator D, to maximize the probability of D making a mistake
- There is a discriminator D that estimates the probability that a sample came from the real data set rather than G and tries to distinguish them



The Formulation of GANs

$$\min_G \max_D V(D,G)$$

$$V(D,G) = \mathrm{E}_{x \sim p_{data}(x)}[\log D(x)] + \mathrm{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))]$$
$$= \mathrm{E}_{x \sim p_{data}(x)}[\log D(x)] + \mathrm{E}_{x \sim p_g(x)}[\log(1 - D(x))]$$

For G fixed, the optimal D is

$$D^* = \max_D V = \max_D \int_x p_{data}(x) \log(D(x)) + p_g(x) \log(1 - D(x))\, dx$$
$$= \max_D a * \log(D) + b * \log(1 - D)$$
$$= \frac{a}{a+b}, where\, a = p_{data}(x), b = p_g(x)$$

The Formulation of GANs

For D fixed, the optimal G is

$$G^* = \min_G \int_x p_{data}(x)\log(D^*(x)) + p_g(x)\log(1 - D^*(x))\,dx$$

$$= \min_G \int_x p_{data}(x)\log\frac{p_{data}(x)}{p_{data}(x)+p_g(x)}\,dx$$

$$+$$

$$\int_x p_g(x)\log\left(\frac{p_g(x)}{p_{data}(x)+p_g(x)}\right)dx$$

$$= \min_G \log\frac{1}{2} + D_{KL}(P_{data}\|M) + \log\frac{1}{2} + D_{KL}(P_g\|M)$$

$$= \min_G -\log 4 + 2\,D_{JS}(P_{data}\|P_g),\ where\ M = \frac{p_{data}+p_g}{2}$$

The Training of GANs

- Training Discriminator

for number of training iterations **do**

for k steps **do**

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
- Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)\right].$$

end for

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

The Training of GANs

- Training Generator

for number of training iterations **do**

for k steps **do**

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
- Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)\right].$$

end for

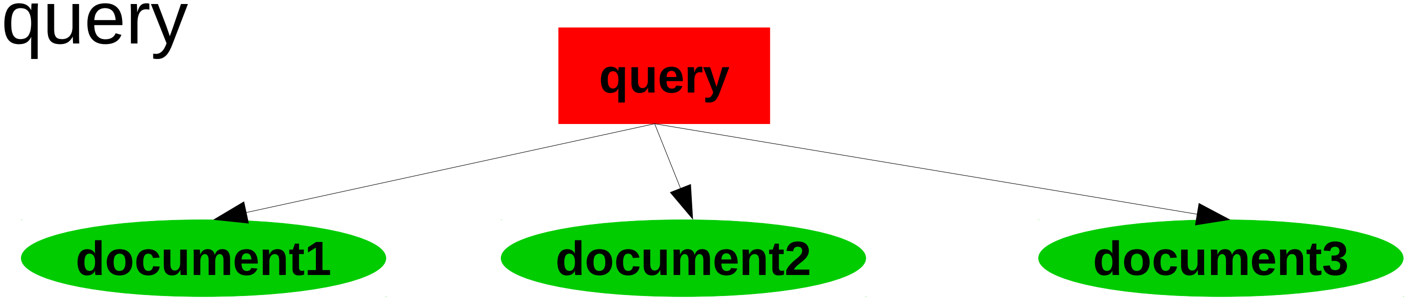
- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

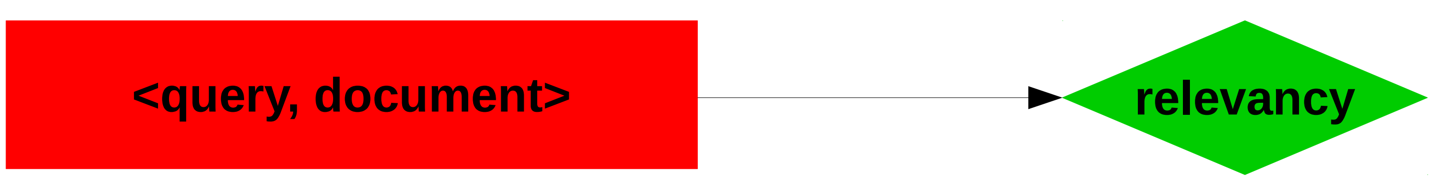
end for

Two Schools of Thinking in Information Retrieval Modeling

- Generative Retrieval
  - Focus on predicting relevant documents given a query



- Discriminative Retrieval
  - Focus on predicting relevancy given a query-document pair



Unify Two Schools of Thinking Based on the Idea of GANs

- Generative Retrieval
  - The generative retrieval model can learn to fit the underlying relevance distribution over documents with the help of discriminator
- Discriminative Retrieval
  - The discriminative retrieval model can not only exploit the true data but also exploit the fake data selected from the generative model to get a better estimation

The Formulation of IRGAN

- Problem Definition
  - Given a set of queries $\{q_1, q_{2,} \ldots, q_N\}$ and a set of documents $\{d_1, d_{2,} \ldots, d_M\}$. A query here can be any specific form of the user's information need such as search keywords, a user profile, or a question
  - The underlying true relevance distribution $p_{true}(d|q,r)$ describes the user's preference distribution over the candidate documents with respect to her submitted query

The Formulation of IRGAN

- Problem Definition
  - Training set
    - Given a set of samples from $p_{true}(d|q,r)$ as the training data
  - Generative Retrieval Model
    - $p_{\theta}(d|q,r)$ which generates/selects relevant documents from the candidate pool for the query q, trying to approximate the true relevance distribution
  - Discriminative Retrieval Model
    - $f_{\phi}(q,d)$ which tries to distinguish between true and fake

The Formulation of IRGAN

- Overall Objective

$$J^{G^*, D^*} = \min_{\theta} \max_{\phi} \sum_{n=1}^{N} \Big(\mathbb{E}_{d \sim p_{\text{true}}(d|q_n, r)} \left[\log D(d|q_n) \right] + \mathbb{E}_{d \sim p_{\theta}(d|q_n, r)} \left[\log(1 - D(d|q_n)) \right] \Big)$$

where

$$D(d|q) = \sigma(f_{\phi}(d, q)) = \frac{\exp(f_{\phi}(d, q))}{1 + \exp(f_{\phi}(d, q))}$$

The Formulation of IRGAN

- Optimizing Discriminative Retrieval

$$\phi^* = \arg\max_{\phi} \sum_{n=1}^{N} \Big(\mathbb{E}_{d \sim p_{\text{true}}(d|q_n,r)} \left[\log(\sigma(f_\phi(d,q_n)) \right] + \mathbb{E}_{d \sim p_{\theta^*}(d|q_n,r)} \left[\log(1 - \sigma(f_\phi(d,q_n))) \right] \Big)$$

The Formulation of IRGAN

- Optimizing Generative Retrieval

$$\theta^* = \arg\min_{\theta} \sum_{n=1}^{N} \Big(\mathbb{E}_{d \sim p_{\text{true}}(d|q_n, r)} \left[\log \sigma(f_\phi(d, q_n)) \right] + \mathbb{E}_{d \sim p_\theta(d|q_n, r)} \left[\log(1 - \boxed{\sigma(f_\phi(d, q_n))}) \right] \Big)$$

$$= \arg\max_{\theta} \sum_{n=1}^{N} \underbrace{\mathbb{E}_{d \sim p_\theta(d|q_n, r)} \left[\log(1 + \exp(f_\phi(d, q_n))) \right]}_{\text{denoted as } J^G(q_n)}$$

$$\boxed{\sigma(f_\phi(d, q))} = \frac{\exp(f_\phi(d, q))}{1 + \exp(f_\phi(d, q))}$$

The Formulation of IRGAN

- Optimizing Generative Retrieval
  - As the sampling of d is discrete, it cannot be directly optimized by SGD from the original GANs
    - 离散数据无法直接反向传播，一般采用 RL 中的 policy gradient 或者 gumbel softmax 来近似计算
    - 离散数据和连续数据 embedding 空间之间存在差异，embedding 上的一点微小改变，很难直接反映到离散数据上
    - The view of Ian Goodfellow:

The Formulation of IRGAN

- The View of Ian Goodfellow:
  - You can make slight changes to the synthetic(合成) data only if it is based on continuous numbers. If it is based on discrete numbers, there is no way to make a slight change.
    - For example, if you output an image with a pixel value of 1.0, you can change that pixel value to 1.0001 on the next step.
    - If you output the word “penguin”(企鹅), you can’t change that to “penguin + .001” on the next step, because there is no such word as “penguin + .001”. You have to go all the way from “penguin” to “ostrich”(鸵鸟).
- Policy gradient based reinforcement learning is used here

The Formulation of IRGAN

- Optimizing Generative Retrieval

$$
\begin{aligned}
&\nabla_\theta J^G(q_n) \\
&= \nabla_\theta \mathbb{E}_{d\sim p_\theta(d|q_n,r)}\left[\log(1+\exp(f_\phi(d,q_n)))\right] \\
&= \sum_{i=1}^{M} \boxed{\nabla_\theta p_\theta(d_i|q_n,r)} \log(1+\exp(f_\phi(d_i,q_n))) \\
&= \sum_{i=1}^{M} \boxed{p_\theta(d_i|q_n,r)\nabla_\theta \log p_\theta(d_i|q_n,r)} \log(1+\exp(f_\phi(d_i,q_n))) \\
&= \mathbb{E}_{d\sim p_\theta(d|q_n,r)}\left[\nabla_\theta \log p_\theta(d|q_n,r)\log(1+\exp(f_\phi(d,q_n)))\right] \\
&\simeq \frac{1}{K}\sum_{k=1}^{K} \nabla_\theta \log p_\theta(d_k|q_n,r) \boxed{\log(1+\exp(f_\phi(d_k,q_n)))},
\end{aligned}
$$

$$\boxed{\nabla_\theta \log(p_\theta) = \frac{1}{p_\theta}\nabla_\theta p_\theta}$$

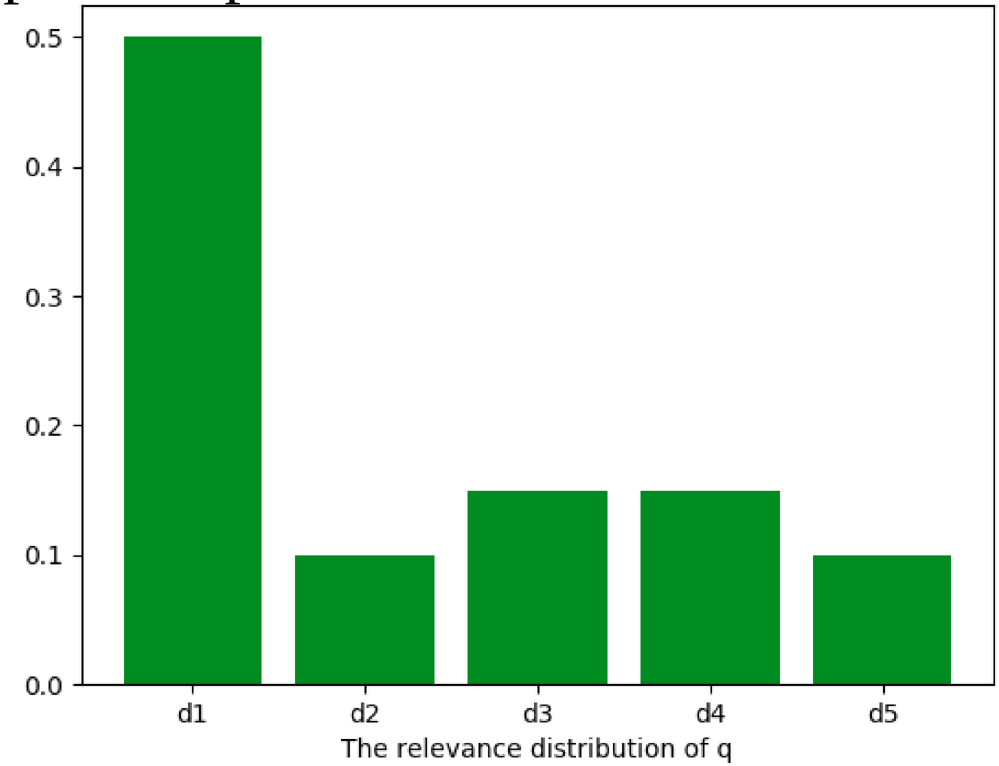
Reward Term

Demonstrate of Policy Gradient

- $G(d) = P_\theta(d|q), where\, d \sim \{d_1, d_2, d_3, d_4, d_5\}$
- $$P(d_i) = \frac{\exp(f(d_i))/\tau}{\sum_j \exp(f(d_j))/\tau}$$

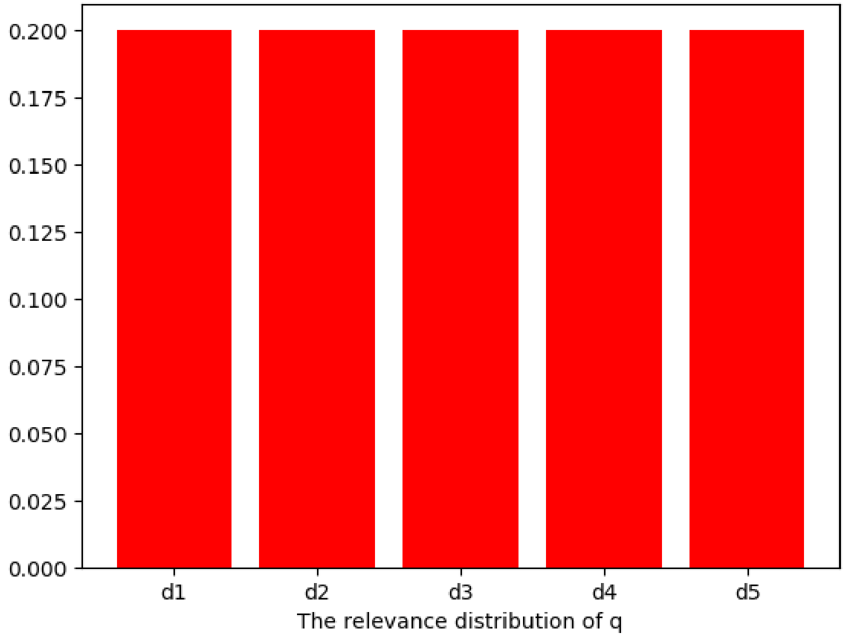
$where\, f$ is the function with respect to specific task

- The Relevance Distribution of q



Demonstrate of Policy Gradient

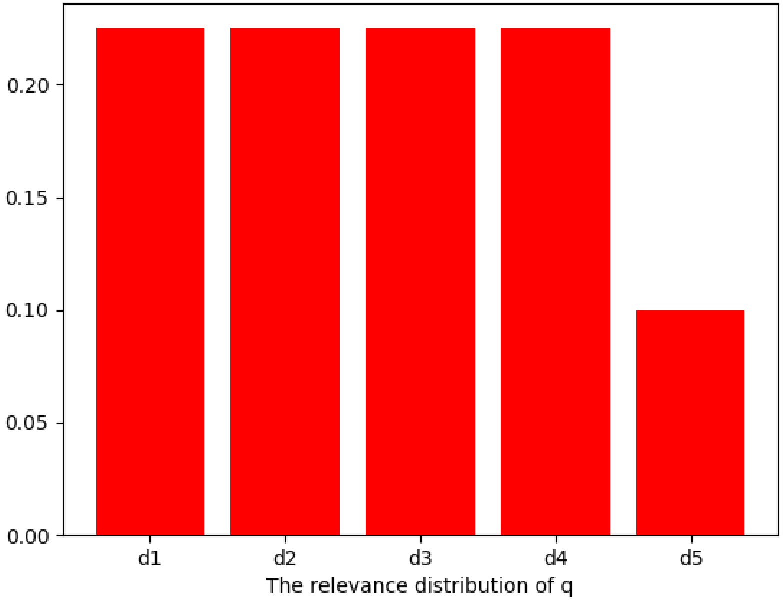
- Step 1:initialize θ



- Step 2:Generate d5

  Observe *negative* reward from D

- Step 3:update θ by policy gradient

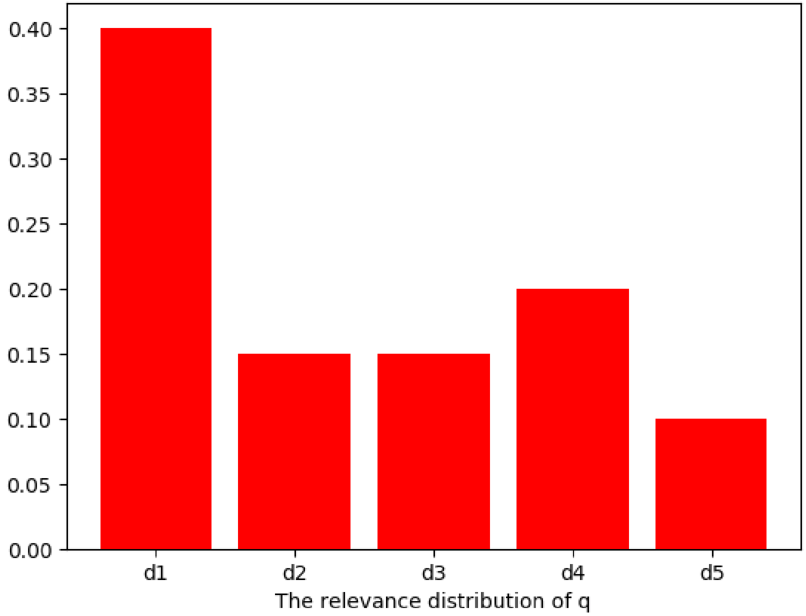


- Step 4:Generate d1

  Observe *positive* reward from D

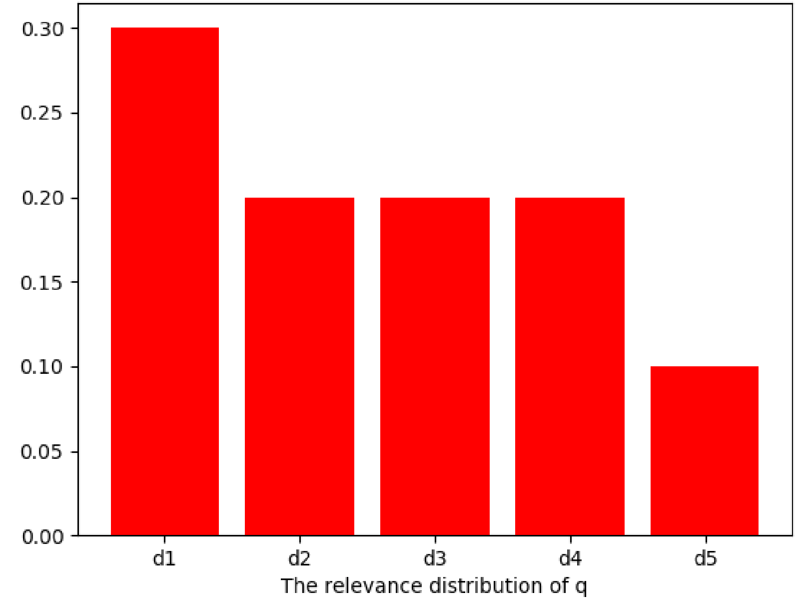
Demonstrate of Policy Gradient

- Step 7:update θ by policy gradient



- Step 8……

- Step 5:update θ by policy gradient



- Step 6:Generate d1

  Observe *positive* reward from D

The Algorithm of IRGAN

Algorithm 1 Minimax Game for IR (a.k.a IRGAN)

Input: generator $p_\theta(d|q,r)$; discriminator $f_\phi(\boldsymbol{x}_i^q)$;
training dataset $\mathcal{S} = \{\boldsymbol{x}\}$

Initialise $p_\theta(d|q,r)$, $f_\phi(q,d)$ with random weights θ, ϕ.
Pre-train $p_\theta(d|q,r)$, $f_\phi(q,d)$ using $\mathcal{S}$
repeat
    for g-steps **do**
        $p_\theta(d|q,r)$ generates K documents for each query q
        Update generator parameters via policy gradient Eq. (5)
    end for
    for d-steps **do**
        Use current $p_\theta(d|q,r)$ to generate negative examples and combine with given positive examples $\mathcal{S}$
        Train discriminator $f_\phi(q,d)$ by Eq. (3)
    end for
until IRGAN converges

Application on Item Recommendation

$$p_\theta(d|q,r) = \frac{\exp(\boxed{g_\theta(q,d)}/\tau)}{\sum_{j\in\mathcal{I}} \exp(g_\theta(q,d)/\tau)}$$

$$D(d|q) = \sigma(f_\phi(d,q)) = \frac{\exp(\boxed{f_\phi(d,q)})}{1+\exp(f_\phi(d,q))}$$

$$g_\theta(q,d) = s_\theta(q,d) \quad \text{and} \quad f_\phi(q,d) = s_\phi(q,d)$$

$$s(u,i) = b_i + \boldsymbol{v}_u^\top \boldsymbol{v}_i$$

Experiment to Item Recommendation

Table 3: Item recommendation results (Movielens).

	P@3	P@5	P@10	MAP
MLE	0.3369	0.3013	0.2559	0.2005
BPR [35]	0.3289	0.3044	0.2656	0.2009
LambdaFM [45]	0.3845	0.3474	0.2967	0.2222
IRGAN-pointwise	0.4072	0.3750	0.3140	0.2418
Impv-pointwise	5.90%*	7.94%*	5.83%*	8.82%*
	NDCG@3	**NDCG@5**	**NDCG@10**	**MRR**
MLE	0.3461	0.3236	0.3017	0.5264
BPR [35]	0.3410	0.3245	0.3076	0.5290
LambdaFM [45]	0.3986	0.3749	0.3518	0.5797
IRGAN-pointwise	0.4222	0.4009	0.3723	0.6082
Impv-pointwise	5.92%*	6.94%*	5.83%*	4.92%*

Table 4: Item recommendation results (Netflix).

	P@3	P@5	P@10	MAP
MLE	0.2941	0.2945	0.2777	0.0957
BPR [35]	0.3040	0.2933	0.2774	0.0935
LambdaFM [45]	0.3901	0.3790	0.3489	0.1672
IRGAN-pointwise	0.4456	0.4335	0.3923	0.1720
Impv-pointwise	14.23%*	14.38%*	12.44%*	2.87%*
	NDCG@3	**NDCG@5**	**NDCG@10**	**MRR**
MLE	0.3032	0.3011	0.2878	0.5085
BPR [35]	0.3077	0.2993	0.2866	0.5040
LambdaFM [45]	0.3942	0.3854	0.3624	0.5857
IRGAN-pointwise	0.4498	0.4404	0.4097	0.6371
Impv-pointwise	14.10%*	14.27%*	13.05%*	8.78%*

The End

Thanks for Listening