Variational factorization machines for preference elicitation in large-scale recommender systems

Jill-Jênn Vie[1] Tomas Rigaux[1] Hisashi Kashima[2]

[1] Inria Saclay, SODA team

[2] Kyoto University

Outline

Preference elicitation

- Getting new info from new users is hard
- We need side information and to model uncertainty

Factorization Machines (FMs)

- FMs are a generalization of latent factor models (Rendle, 2012)
- Used for both regression and classification
- Sometimes better than their deep counterparts

In this paper

- Variational Factorization Machines
- Variational: Bayesian inference $\rightarrow$ optimization

Recommender Systems as Matrix Completion

Problem

- Every user rates few items (1 %)
- How to infer missing ratings?

Example

Satoshi	?	5	2	?
Kasumi	4	1	?	5
Takeshi	3	3	1	4
Joy	5	?	2	?

Recommender Systems as Matrix Completion

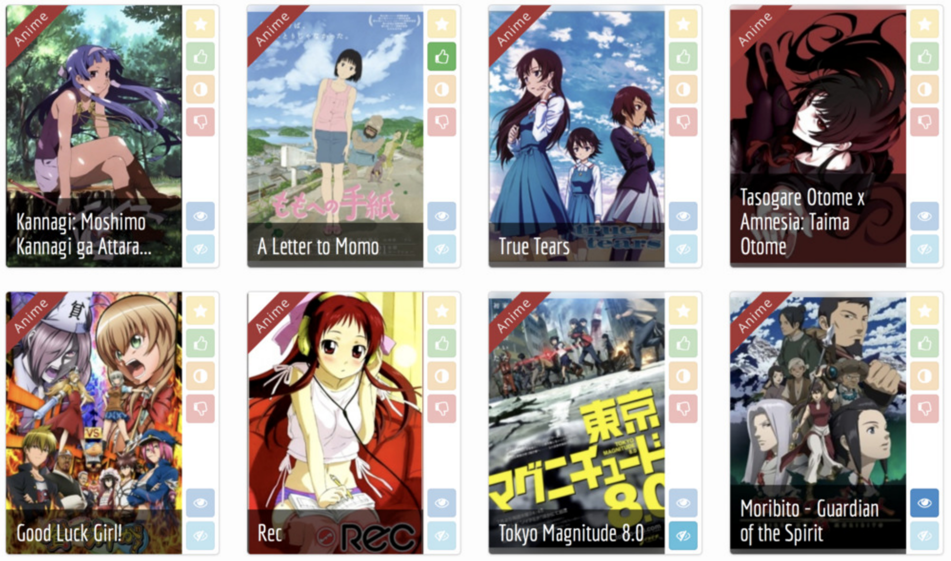
Problem

- Every user rates few items (1 %)
- How to infer missing ratings?

Example

Satoshi	3	5	2	2
Kasumi	4	1	4	5
Takeshi	3	3	1	4
Joy	5	2	2	5

Preference Elicitation: select an informative batch of K items



Preference Elicitation: learn user embeddings in latent space

Matrix factorization for collaborative filtering

Approximate R ratings $n \times m$ by learning embeddings for user and item

$$\left.\begin{array}{r} U \text{ user embeddings } n \times d \\ V \text{ item embeddings } m \times d \end{array}\right\} \text{such that } R \simeq UV^T$$

Fit

Learn U and V to minimize $||R - UV^T||_2^2 + \lambda \cdot \text{regularization}$

Predict: Will user i like item j?

Just compute $\langle \boldsymbol{u}_i, \boldsymbol{v}_j \rangle$

The actual model also contains bias terms for user i and item j

$$r_{ij} = \mu + w_i^u + w_j^v + \langle \boldsymbol{u}_i, \boldsymbol{v}_j \rangle$$

How to model side information?

If you know user i watched item j at the cinema (or on TV, or on smartphone), how to model it?

r_{ij}: rating of user i on item j

Collaborative filtering

$$r_{ij} = w_{\text{user } i} + w_{\text{item } j} + \langle \boldsymbol{v}_{\text{user } i}, \boldsymbol{v}_{\text{item } j} \rangle$$

How to model side information?

If you know user i watched item j at the cinema (or on TV, or on smartphone), how to model it?

r_{ij}: rating of user i on item j

Collaborative filtering

$$r_{ij} = w_{\text{user } i} + w_{\text{item } j} + \langle \boldsymbol{v}_{\text{user } i}, \boldsymbol{v}_{\text{item } j} \rangle$$

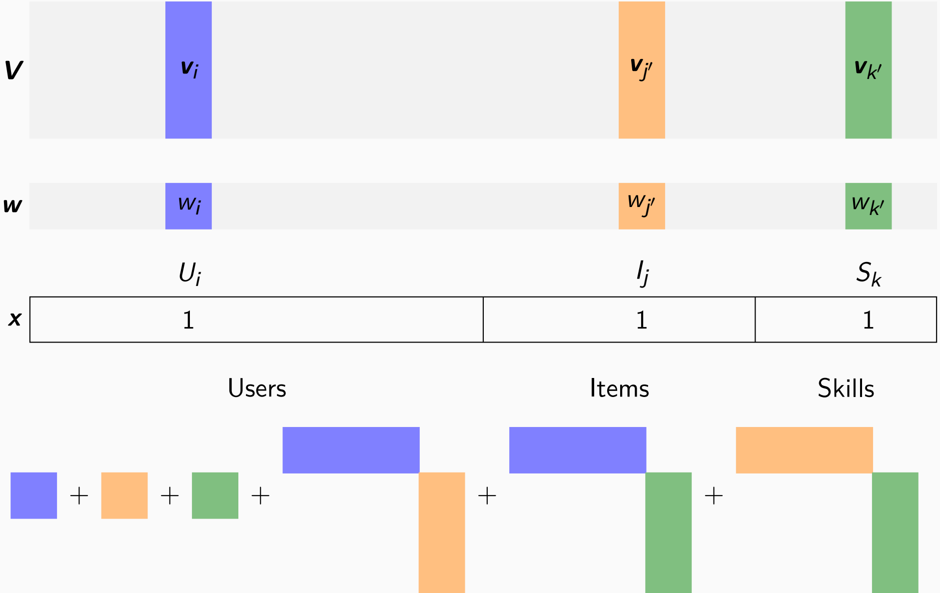
With side information

$$r_{ij} = w_{\text{user } i} + w_{\text{item } j} + w_{cinema} + \langle \boldsymbol{v}_{\text{user } i}, \boldsymbol{v}_{\text{item } j} \rangle + \langle \boldsymbol{v}_{\text{user } i}, \boldsymbol{v}_{\text{cinema}} \rangle + \langle \boldsymbol{v}_{\text{item } j}, \boldsymbol{v}_{\text{cinema}} \rangle$$

Encoding the problem using sparse features

Users			Items				Formats		
U_1	U_2	U_3	I_1	I_2	I_3	I_4	cinema	TV	mobile
0	1	0	0	1	0	0	0	1	0
0	0	1	0	0	1	0	0	1	0
0	1	0	0	0	1	0	1	0	0
0	1	0	0	1	0	0	1	0	0
1	0	0	0	0	0	1	0	1	0

Graphically: factorization machines



Formally: factorization machines

Learn bias w_k and embedding $\boldsymbol{v_k}$ for each feature k such that:

$$y(\boldsymbol{x}) = \mu + \underbrace{\sum_{k=1}^{K} w_k x_k}_{\text{linear regression}} + \underbrace{\sum_{1 \leq k < l \leq K} x_k x_l \langle \boldsymbol{v_k}, \boldsymbol{v_l} \rangle}_{\text{pairwise interactions}}$$

This model is for regression

If classification, use a link function like softmax/sigmoid or Gaussian CDF

Steffen Rendle. "Factorization Machines with libFM". In: *ACM Transactions on Intelligent Systems and Technology (TIST)* 3.3 (2012), 57:1–57:22. DOI: 10.1145/2168752.2168771

Training using, for example, SGD

Take a batch $(\boldsymbol{X}_B, y_B)$ and update the parameters such that the error is minimized.

- Loss in classification: cross-entropy
- Loss in regression: squared error

Algorithm 1 SGD

---

for batch $\boldsymbol{X}_B, y_B$ **do**
  for k feature involved in this batch $\boldsymbol{X}_B$ **do**
    Update $w_k, \boldsymbol{v}_k$ to decrease loss estimate $\mathcal{L}$ on $\boldsymbol{X}_B$
  end for
end for

Why do we prefer distributions over point estimates?

- Because we can measure uncertainty
- More robust for critical applications
- Can guide sequential estimation (preference elicitation)

Variational inference

Approximate true posterior with an easier distribution (Gaussian)

Idea: increase the ELBO $\Rightarrow$ increase the objective

$$\log p(\boldsymbol{y}) \geq \underbrace{\sum_{i=1}^{N} \mathbb{E}_{q(\theta)}[\log p(y_i|x_i,\theta)] - \mathrm{KL}(q(\theta)||p(\theta))}_{\text{Evidence Lower Bound (ELBO)}}$$

$$= \sum_{i=1}^{N} \mathbb{E}_{q(\theta)}[\log p(y_i|x_i,\theta)] - \mathrm{KL}(q(w_0)||p(w_0)) - \sum_{k=1}^{K} \mathrm{KL}(q(\theta_k)||p(\theta_k))$$

Needs to be rescaled for mini-batching (see in the paper)

Variational inference

$$\text{Priors } p(w_k) = \mathcal{N}(\nu^w_{g(k)}, 1/\lambda^w_{g(k)}) \qquad p(v_{kf}) = \mathcal{N}(\nu^{v,f}_{g(k)}, 1/\lambda^{v,f}_{g(k)})$$

$$\text{Approx. posteriors } q(w_k) = \mathcal{N}(\mu^w_k, (\sigma^w_k)^2) \qquad q(v_{kf}) = \mathcal{N}(\mu^{v,f}_k, (\sigma^{v,f}_k)^2)$$

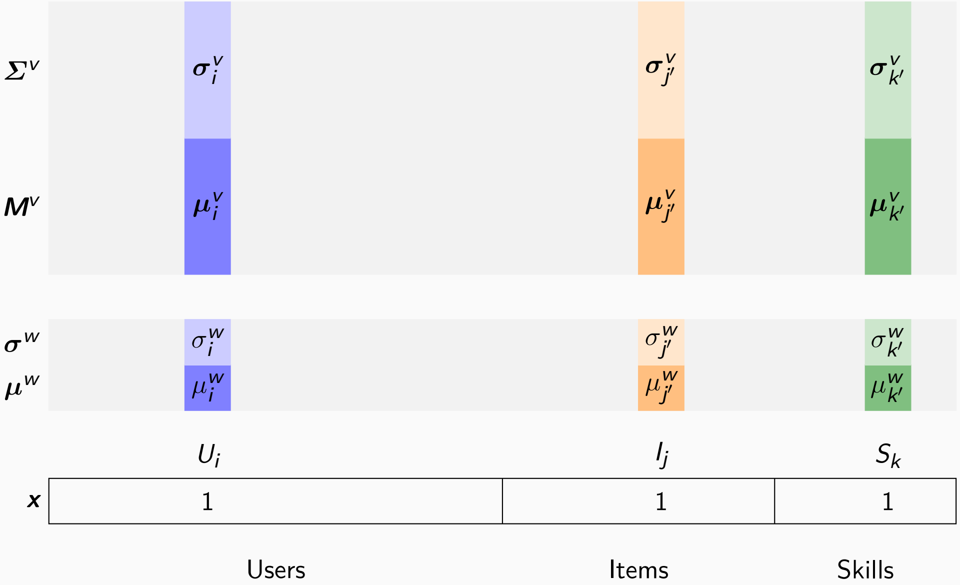
Idea: increase the ELBO $\Rightarrow$ increase the objective

$$\log p(\boldsymbol{y}) \geq \sum_{i=1}^N \underbrace{\mathbb{E}_{q(\theta)}[\log p(y_i|x_i,\theta)] - \mathrm{KL}(q(\theta)||p(\theta))}_{\text{Evidence Lower Bound (ELBO)}}$$

$$= \sum_{i=1}^N \mathbb{E}_{q(\theta)}[\log p(y_i|x_i,\theta)] - \mathrm{KL}(q(w_0)||p(w_0)) - \sum_{k=1}^K \mathrm{KL}(q(\theta_k)||p(\theta_k))$$

Needs to be rescaled for mini-batching (see in the paper)

Graphically: Variational Factorization Machines



VFM training

Algorithm 2 Variational Training (SGVB) of FMs

for each batch $B \subseteq \{1, \ldots, N\}$ **do**
    Sample $w_0 \sim q(w_0)$
    for $k \in F(B)$ feature involved in batch B **do**
        Sample S times $w_k \sim q(w_k)$, $\boldsymbol{v}_k \sim q(\boldsymbol{v}_k)$
    end for
    for $k \in F(B)$ feature involved in batch B **do**
        Update parameters $\mu_k^w, \sigma_k^w, \boldsymbol{\mu}_k^v, \boldsymbol{\sigma}_k^v$ to increase ELBO estimate
    end for
    Update hyper-parameters $\mu_0, \sigma_0, \nu, \lambda, \alpha$
    Keep a moving average of the parameters to compute mean predictions
end for

Then σ can be reused for preference elicitation (see how in the paper)

Stochastic weight averaging

A beneficial regularization: keep all weights over training epochs and average them.

Connections to Polyak-Ruppert averaging, aka stochastic weight averaging

Experiments on real data

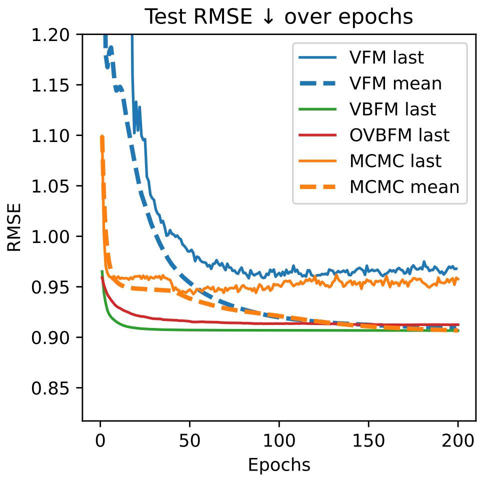
Task	Dataset	#users	#items	#entries	Sparsity
Regression	movie100k	944	1683	100000	0.937
	movie1M	6041	3707	1000209	0.955
Classification	movie100	100	100	10000	0
	movie100k	944	1683	100000	0.937
	movie1M	6041	3707	1000209	0.955
	Duolingo	1213	2416	1199732	0.828

Models

- The proposed approach VFM
- libFM MCMC implementation
- We found another preprint VBFM [2] only for regression

Results on regression

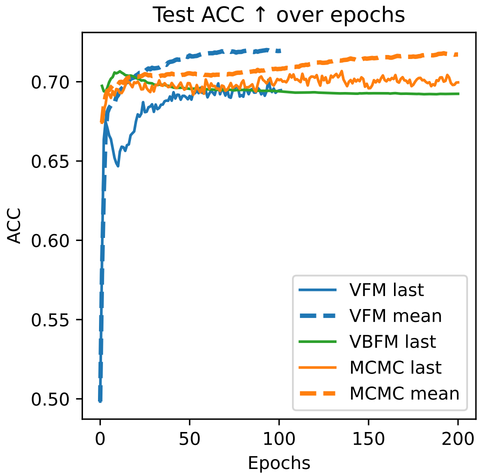
RMSE	Movie100k	Movie1M
MCMC	**0.906**	**0.840**
VFM	**0.906**	0.854
VBFM	0.907	0.856
OVBFM	0.912	0.846



OVBFM is online (batch size = 1) of VBFM

Results on classification

ACC	Movie100k	Movie1M	Duolingo
MCMC	0.717	0.739	**0.848**
VFM	**0.722**	**0.746**	0.846
VBFM	0.692	0.732	0.842



In the paper, we also report AUC and mean average precision.

Conclusion

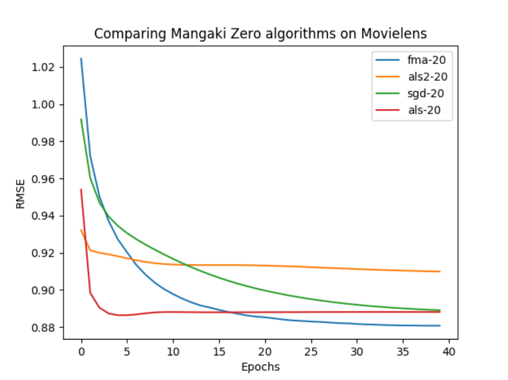
- FMs are a strong baseline
- In this paper we present a variational approach for learning them
  - so that we can deal with u n c e r t a i n t y
- Our method is batched so suitable for large-scale datasets
- We have better performance on some (not all) classification datasets; perhaps due to Adam optimizer or stochastic weight averaging (beneficial regularization)

Thanks for listening!

VFM is implemented in TF & PyTorch

$\mathbb{E}_{q(\theta)}[\log p(y_i | \boldsymbol{x}_i, \theta)]$ becomes `outputs.log_prob(observed).mean()`
Same implementation for classification and regression: the only difference in the distribution (Bernoulli vs. Gaussian)

Feel free to try it on GitHub (`vfm.py`):
github.com/jilljenn/vae



See more benchmarks on
github.com/mangaki/zero

[1] Steffen Rendle. “Factorization Machines with libFM”. In: *ACM Transactions on Intelligent Systems and Technology (TIST)* 3.3 (2012), 57:1–57:22. DOI: 10.1145/2168752.2168771.

[2] Avijit Saha et al. “Scalable Variational Bayesian Factorization Machine”. 2017.