Variational Factorization Machines for Preference Elicitation in Large-Scale Recommender Systems

Jill-Jênn Vie
Inria Saclay – SODA
Palaiseau, France
jill-jenn.vie@inria.fr

Tomas Rigaux
Inria Saclay – SODA
Palaiseau, France
tomas.rigaux@inria.fr

Hisashi Kashima
Kyoto University
Kyoto, Japan
kashima@i.kyoto-u.ac.jp

Abstract—**Factorization machines (FMs) are a powerful tool for regression and classification in the context of sparse observations, that has been successfully applied to collaborative filtering, especially when side information over users or items is available. Bayesian formulations of FMs have been proposed to provide confidence intervals over the predictions made by the model, however they usually involve Markov-chain Monte Carlo methods that require many samples to provide accurate predictions, resulting in slow training in the context of large-scale data. In this paper, we propose a variational formulation of factorization machines that allows us to derive a simple objective that can be easily optimized using standard mini-batch stochastic gradient descent, making it amenable to large-scale data. Our algorithm learns an approximate posterior distribution over the user and item parameters, which leads to confidence intervals over the predictions. We show, using several datasets, that it has comparable or better performance than existing methods in terms of prediction accuracy, and provide some applications in active learning strategies, e.g., preference elicitation techniques.**

I. Introduction

Data collected from a crowd is imperfect: ratings in recommender systems, outcomes of students over educational exercises, annotations provided by humans for crowdsourcing tasks. It is crucial to model it properly in order to provide interesting recommendations to users, identify misconceptions of students, determine the true labels. The challenge here relies in modeling both the items being annotated, and the users annotating them.

In the particular case of recommender systems, we have access to some ratings provided by users over items, and we want to generalize to new user-item pairs. Collaborative filtering [1] is a famous technique that consists in learning an embedding for each user and item, so that the rating can be expressed as a function of the user and item embeddings. In various applications, side information is also available on either users or items, such as: the different genres of the movies, or their actors, directors, etc. The challenge now becomes, how to use this extra information to improve the recommendations? This is why factorization machines were developed [2]. They generalize collaborative filtering in the presence of side information.

Models typically encountered in collaborative filtering, such as the latent factor model, learn point estimates of the user and item parameters; but it is sometimes more useful to learn distributions over these parameters, in order to have confidence intervals over predictions that can guide decision making. In educational applications, this is particularly useful: what is the uncertainty over the algebra level of this student? Should I continue to ask them algebra questions, or should I switch to geometry? On a recommender system, knowing that the predictions are uncertain for a specific user-item pair can be more valuable than just predicting an average grade; using this uncertainty, one can even let the user control whether they want recommendations with high confidence, or more risky ones in order to explore more the item database. This is particularly important when new users come in, and that we have to identify in few questions their position in latent space to quickly provide good recommendations to them. This is called preference elicitation [3]. One way to do so is to use probabilistic matrix factorization [4].

However, when we learn these models using Bayesian inference, the posterior distribution over the parameters is usually intractable, and we resort to MCMC techniques like Metropolis-Hastings [5] or Gibbs sampling [6], [7]. These techniques have the advantage to recover the exact distribution asymptotically, at the cost of many samples. Variational inference [8], [9] has been proposed to compute at a cheaper cost an approximate inference of the posterior.

In this paper, we propose a variational approach to learn factorization machines. Distributions over the user and item parameters are estimated easily, and likelihood maximization is done by increasing a lower bound. By acting directly on factorization machines, our results naturally extend to the latent factor models typically encountered in collaborative filtering such as probabilistic matrix factorization or item response theory.

Our contribution is a simple algorithm for learning Bayesian factorization machines that works both for classification and regression tasks. We show using several datasets of various sparsity that it achieves comparable performance at predicting unseen pairs than existing methods for regression tasks, and better performance in classification tasks.

This paper is built as follows. First, we expose related work, then we present factorization machines, our variational objective, our algorithm, experiments and results. In particular, we show how those VFMs can be used for querying the preference of users interactively, which is the goal of large-scale recommender systems or intelligent tutoring systems.

II. Related Work

Factorization machines (FMs) are a multilinear model of regression or classification using sparse features, described in [2]. Under their regression form, FMs can be seen as a generalization of the latent factor model encountered in collaborative filtering. Under their classification form, Vie and Kashima [10] have shown that FMs can be seen as a generalization of multidimensional item response theory [11]. Variational learning of item response theory models have been proposed at EDM 2020 [12] and the methods in the present article can be directly applied to learning multidimensional item response theory models.

Rendle [2] describes several algorithms to train factorization machines implemented in C++ in the libFM package, notably a Bayesian version where every feature (user, item, or other) has a Gaussian prior, and Gamma hyperpriors. Using a Markov-chain Monte Carlo (MCMC) method called Gibbs sampling, they can sample the hyperpriors, the parameters, then the predictions. One specific advantage of MCMC is that it automatically optimizes the regularization parameter during training, avoiding to perform costly hyper-parameter selection through cross-validation.

Since then, several variants of factorization machines have been proposed, such as deep factorization machines [13], [14], convex factorization machines [15] and higher-order factorization machines [16]. Higher-order FMs consider not only pairwise interactions but n-order terms. In all these works, the MCMC version is recognized as a hard baseline, that sometimes beats deep counterparts [17], [18], which is why it is our principal competitor in this paper.

To the best of our knowledge, the only independent attempts at developing variational learning for factorization machines are an open-source implementation in Chainer [19] reaching higher RMSE on Movielens 1M than our method; and an unpublished preprint for the regression task only [20], with very efficient open-source code in C++, for either full batch or online scenario (batch size of one). As FMs are multilinear in their parameters, [20] can compute the ELBO objective in closed form, the exact gradients and the best possible updates for each parameter, so their approach is similar to alternating least squares. Terms are cached to optimize the complexity of computation. However, in the classification task, there is no closed form of the objective, for our choice of distributions and link function.

III. Factorization Machines

Let $d \geq 0$ be an integer. Every feature $k = 1, \ldots, K$, be it a user, an item, a tag, etc. is parameterized using one scalar bias $w_k \in \mathbb{R}$ and an embedding $\boldsymbol{v}_k \in \mathbb{R}^d$. We will note V the matrix having all embeddings $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_K$ as columns.

Each sample $i = 1, \ldots, n$ is described by a pair $(\boldsymbol{x}, y)$ where y is a label (say, a rating) and $\boldsymbol{x}$ is a sparse vector which determines the presence ($x_k \neq 0$) or absence ($x_k = 0$) of each feature k, see Table I. For example, if there are N_u users, N_i items and no side information, one can encode the event "user i gave item j 5 stars" with the pair

	Sparse features $\boldsymbol{x}$						$\boldsymbol{y}$
	U_0	U_1	U_2	I_0	I_1	I_2	
User 1 Item 1: 5/5	0	1	0	0	1	0	5
User 1 Item 2: 4/5	0	1	0	0	0	1	4
User 2 Item 1: 2/5	0	0	1	0	1	0	2
User 2 Item 1: 2/5	0	0	1	0	1	0	2
User 2 Item 2: 5/5	0	0	1	0	0	1	5

TABLE I: Example of sparse features encoded for a factorization machine. Each column of $\boldsymbol{x}$ corresponds to either a user or an item.

$$\begin{cases} \boldsymbol{x} = (0, \ldots, 0, \underbrace{1}_{i}, 0, \ldots, 0, \underbrace{1}_{N_u+j}, 0, \ldots, 0), \\ y = 5. \end{cases}$$

The expression at the core of factorization machines is the following, see also Figure 1:

$$\begin{aligned} y(\boldsymbol{x}) &= w_0 + \sum_{k=1}^{K} w_k x_k + \sum_{1 \leq k < \ell \leq K} x_k x_\ell \langle \boldsymbol{v}_k, \boldsymbol{v}_\ell \rangle \\ &= w_0 + \langle \boldsymbol{w}, \boldsymbol{x} \rangle + \frac{1}{2} \left(||V\boldsymbol{x}||_2^2 - \sum_{f=1}^{d} ||(V^T)_f \circ \boldsymbol{x}||_2^2 \right) \\ &= w_0 + \langle \boldsymbol{w}, \boldsymbol{x} \rangle + \frac{1}{2} \left(||V\boldsymbol{x}||_2^2 - ||(V \circ V)^T (\boldsymbol{x} \circ \boldsymbol{x})||_2^2 \right) \end{aligned}$$

where $\circ$ denotes the pairwise product and w_0 is a global bias. Therefore the parameters of the model are $(w_0, \boldsymbol{w}, V)$. The last expression is easier to compute as $\boldsymbol{x}$ is sparse, which also makes it convenient for computing the predictions of one batch.

In the particular case where there is no side information, we can recover collaborative filtering: if there are N_u users, N_i items, r_{ij} is the rating given by user i over item j and $\boldsymbol{x}$ is the concatenation of a 1-hot vector of size N_u having 1 at position i, and a 1-hot vector of size N_i having 1 at position j, we get:

$$y(\boldsymbol{x}) = w_0 + w_i + w_{N_u+j} + \langle \boldsymbol{v}_i, \boldsymbol{v}_{N_u+j} \rangle.$$

Therefore, the first N_u entries of w (resp. V) contain the user biases (resp. embeddings), the next N_i entries of w (resp. V) contain the item biases (resp. embeddings).

a) Regression: For a new sample $\boldsymbol{x}$ of sparse features, FM will output a prediction $\hat{y} \in \mathcal{N}(y(\boldsymbol{x}), 1/\alpha)$ where $\alpha > 0$ models noise and can be seen as a regularization parameter, and $\mathcal{N}$ is the normal distribution. While optimizing likelihood, we recover a squared loss term.

b) Classification: For a new sample $\boldsymbol{x}$ of sparse features, FM will output a prediction $\hat{y} \in \mathcal{B}(\phi(y(\boldsymbol{x}))$ where ϕ is a link function and $\mathcal{B}$ is the Bernoulli distribution. In this paper $\phi = \sigma : x \mapsto 1/(1 + \exp(-x))$ is the sigmoid function but in the libFM package, $\phi = \Phi$ is the cumulative density function of the standard normal distribution, because it is more convenient

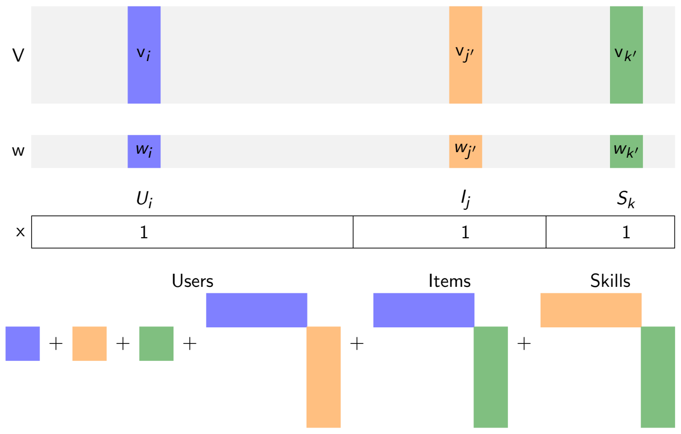

Fig. 1: Architecture of a FM. Pairwise dot products of embeddings are added to the biases of activated features.

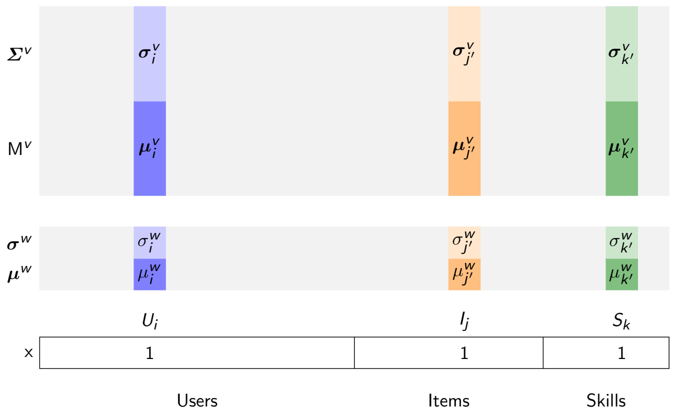

Fig. 2: Parameters of a VFM, the global bias being omitted for clarity. For each feature we learn a distribution over its bias and a distribution over its embedding. Together, they form an approximate posterior distribution from which we can sample the parameters, and the predictions.

for Gibbs sampling, although very similar in shape. Please note that when $d = 0$, the V term vanishes and we recover logistic regression when $\phi = \sigma$ and probit regression when $\phi = \Phi$. There is no regularization parameter here, as explained by [21] (possibly because the prediction never matches the true label in classification scenarios).

A. *Training of FMs*

Training of FMs can be done using standard algorithms such as stochastic gradient descent (SGD), alternating least squares (ALS) in their regression form, or Gibbs sampling [2] in their Bayesian version where parameters w_0, $\boldsymbol{w}$ and V are random variables. In libFM, the lowest error is usually achieved by training FMs using MCMC.

IV. Variational Factorization Machines (VFM)

We now detail the variational training of FMs which is our main contribution in this paper.

Let θ denote the random variables of a VFM, i.e. w_0, $\boldsymbol{w}$ and V. We will denote sample i by its features $\boldsymbol{x}_i$ and outcome y_i.

For regression, the likelihood is $p(y_i|\boldsymbol{x}_i, \theta) = \mathcal{N}(y(\boldsymbol{x}_i), 1/\alpha)$ while for classification, the likelihood is $p(y_i|\boldsymbol{x}_i, \theta) = \mathcal{B}(\sigma(y(\boldsymbol{x}_i))$ where σ is the sigmoid function.

A. *Priors*

We use the same prior as libFM:

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

where $g(k) > 0$ is an integer representing the group of feature k (user, item, etc.) and $g(0) = 0$ for the global bias. Rendle uses hyperpriors $\nu \sim \mathcal{N}(0, 1), \lambda \sim \Gamma(1, 1)$ where Γ denotes the Gamma distribution because it is convenient for Gibbs sampling as it is the conjugate of the normal distribution [2]. In our case, we just treat ν^w, λ^w, $\nu^{v,f}$ and $\lambda^{v,f}$ as parameters.

We found similar results with simpler priors such as $\mathcal{N}(0, 1)$. Saha et al. use $\nu^w = \nu^{v,f} = 0$ [20].

B. *Approximate posteriors*

Our variational approximation:

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

where $(\mu^w_k, \sigma^w_k, \mu^{v,f}_k, \sigma^{v,f}_k)$ are parameters that we will learn. Hence, we have $2(d+1)(K+G)+5$ parameters to estimate representing respectively the posterior of entities, the hyperparameters related to prior of groups, the global bias, and and α, see Figure 2.

Sampling a random variable $\theta_k \in \{w_k, \boldsymbol{v}_k\}$ from the posterior is easy because we just need to do:

$$\theta_k \leftarrow \mu^\theta_k + \varepsilon \cdot \sigma^\theta_k$$

for some $\varepsilon \sim \mathcal{N}(0, 1)$. This is called the reparameterization trick [9].

C. *Evidence lower bound objective*

What we would like to increase is the likelihood over all training samples:

$$\log p(\boldsymbol{y}) = \sum_{i=1}^N \log p(y_i)$$

But it's intractable. So instead we maximize a lower bound over $\log p(\boldsymbol{y})$:

$$\begin{aligned}\log p(\boldsymbol{y}) &\geq \underbrace{\sum_{i=1}^N \mathbb{E}_{q(\theta)}[\log p(y_i|\boldsymbol{x}_i, \theta)] - \mathrm{KL}(q(\theta)||p(\theta))}_{\text{Evidence Lower Bound (ELBO)}} \\ &= \sum_{i=1}^N \mathbb{E}_{q(\theta)}[\log p(y_i|\boldsymbol{x}_i, \theta)] - \mathrm{KL}(q(w_0)||p(w_0)) \\ &\quad - \sum_{k=1}^K \mathrm{KL}(q(\theta_k)||p(\theta_k))\end{aligned}$$

where $q(\theta_k) = q(w_k)q(\boldsymbol{v}_k) = q(w_k)\prod_{f=1}^d q(v_{kf})$.

Hopefully, by increasing the Evidence Lower Bound (ELBO), we will indirectly increase the log-likelihood. This is at the core of variational inference.

In our case, the ELBO is easy to derive as the KL term is between two Gaussian distributions so it can be computed in closed form. Also all parameters and pairwise independent so everything can be rewritten as a product of univariate Gaussians.

When we estimate the ELBO for a single batch of samples $B \subseteq \{1, \ldots, N\}$, we have to rescale our estimate to alleviate bias. Let N_k (resp. N_k^B) be the number of occurrences of feature k in the training set (resp. batch B). We denote $F(B)$ the set of unique features that are present in the batch: $k \in F(B) \iff N_k^B > 0 \iff \exists i \in B, x_{ik} > 0$.

$$\mathcal{L}_B = \frac{N}{|B|}\mathbb{E}_{q(\theta)}\left[\sum_{i \in B} \log p(y_i|\boldsymbol{x}_i, \theta)\right] - \frac{K}{\sum_k \alpha_k} \sum_{k \in F(X_B)} \alpha_k \mathrm{KL}(q(\theta_k)||p(\theta_k))$$

where $\alpha_k = N_k^B / N_k$ is a rescaling weight for debiasing the ELBO estimate. In practice though, we compute such a weighted combination of KL for each group (users, items) in order to ensure unbiasedness.

The expected log-likelihood can also be written in closed form in the regression case, as the FM model is multilinear in the parameters. This is what [2], [20] do to speed up the computations. However, in the classification case, such closed form is not feasible and we have to resort to sampling. S is the number of variational samples, although in most of this paper, $S = 1$.

$$\begin{aligned}\mathcal{L}_B &= \frac{N}{|B|S} \sum_{s=1}^{S} \sum_{i \in B} \log p(y_i|\boldsymbol{x}_i, \theta^{(s)})] \\ &\quad - \frac{K}{\sum_k \alpha_k} \sum_{k \in F(B)} \alpha_k \mathrm{KL}(q(\theta_k)||p(\theta_k)) \\ &\theta^{(1)}, \ldots, \theta^{(S)} \sim q(\theta) \\ &\iff \theta_k^{(s)} = \mu_k^\theta + \varepsilon \cdot \sigma_k^\theta \quad \varepsilon \sim \mathcal{N}(0, 1)\end{aligned}$$

The gradients of $\mathcal{L}_B$ with respect to parameters $\mu_k^w, \sigma_k^w, \mu_k^{v,f}, \sigma_k^{v,f}$ are easy to compute because the prediction y is linear in them in the regression case, log-linear in the classification case.

We learn the parameters μ, σ and hyper-parameters ν, λ, α using Adam optimizer over minibatches. We only update the parameters of features that are present in the batch, which simplifies computation over one batch. Our algorithm is displayed in Algorithm 1.

D. Evaluating predictions on a test set

There are various ways to use the learned distributions to perform prediction. We can either sample from the last iterate of the learned distribution of w_0, $\boldsymbol{w}$ and V, or use the last mean (what is called `last` in experiments), or average iterates of the distribution parameters (what is called `mean` in the experiments). The latter approach has been denoted as Polyak-Ruppert averaging [22] or stochastic weight averaging and acts as a beneficial regularization [23]–[25].

Algorithm 1 Variational Training 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 $w_k \sim q(w_k)$, $\boldsymbol{v}_k \sim q(\boldsymbol{v}_k)$
            ▷ Only required if we do not have a closed form
    end for
    for $k \in F(B)$ feature involved in batch $B$ do
        Update parameters $\mu_k^w, \sigma_k^w, \mu_k^{v,f}, \sigma_k^{v,f}$ to increase ELBO estimate $\mathcal{L}_B$ for this batch
    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
```

V. Experiments

We conducted experiments on real datasets. Given existing entries, the task consisted in predicting the remaining ones.

A. Tasks and Datasets

The datasets are described in Table II. We distinguish three tasks: regression and classification, which can be respectively understood as matrix completion and binary matrix completion; and preference elicitation.

a) Regression: This task is related to collaborative filtering in recommender systems. Ratings are on an ordinal scale, between 1 and 5 (sometimes including decimal ratings such as 2.5). Predictions on a 20% test set are compared using root mean squared error (RMSE):

$$RMSE = \sqrt{\frac{1}{|D|} \sum_{i \in D} (y_i - \hat{y}_i)^2}.$$

b) Classification: We generated binary Movie100k and Movie1M datasets from the original datasets by setting the outcome to $r_{ij} = 1$ if the rating that user i gave to item j was 4 or 5 stars, 0 otherwise. Predictions on a 20% test set are compared using accuracy, area under the ROC curve (AUC) and mean average precision (MAP). This latter metric summarizes a precision-recall curve as a weighted mean of precisions achieved at each threshold, with the increase in recall from the previous threshold used as weight:

$$MAP = \sum_{n \geq 1} (R_n - R_{n-1}) P_n \quad \sum_{n \geq 1} R_n - R_{n-1} = 1$$

where P_n (resp. R_n) is the precision (resp. recall) at the nth threshold.

Task	Dataset	#users	#items	#entries	Sparsity
Regression	Movie100k	944	1683	100000	0.937
	Movie1M	6041	3707	1000209	0.955
Classification	Movie10k	100	100	10000	0
	Movie100k	944	1683	100000	0.937
	Movie1M	6041	3707	1000209	0.955
	Duolingo	1213	2416	1199732	0.828

TABLE II: Datasets used for our experiments.

We also made experiments using some side information. The Duolingo dataset for second language acquisition modeling[1] was used for a competition at the BEA workshop of the NAACL-HLT conference [14], [26]. It contains the traces of 1,200 English-speaking users learning French by answering exercises among 3 different formats (reverse translate, type what they are hearing, listen and type). The observed outcome of user i learning word j in format of exercise k is 1 if they got it correct, 0 otherwise. Therefore the sparse features are a concatenation of 3 one-hot vectors corresponding to the following groups: user, word, and format of exercise. There are 1.2M million entries but some people try several times the same word over time (only 500k unique user–word pairs), and in this paper we consider the same prediction in this case, as we do not explicitly model learning over time. In order to do it, we can use temporal features as side information, see the literature in knowledge tracing [10], [14].

c) Preference elicitation: For this final task, we want to guess which items the user knows by selecting few items to ask. We generated the Movie10k dataset for classification from Movielens 25M by first selecting the 100 movies with the most ratings, then randomly choosing 100 users that had rated at least one but not all of those movies. The Movie10k dataset is the complete 100×100 binary matrix R whose element r_{ij} corresponds to 1 if user i has rated movie j, 0 otherwise.

80% of users are fully observed in the training set for learning item parameters. Once item parameters have been learned, they are frozen and items are separated into interactive training (50%) and validation sets (50%). For each of the remaining 20% of unobserved users in parallel, we select 4 items in the interactive training set for which we reveal whether they have rated those movies or not. Then we update those users' variational parameters, mean and variance, evaluate the predictions on the validation set using ACC, AUC, MAP like in the classification task, then select 4 items again. Another quantity of interest is the reduction in variance for the predictions of items in the validation set.

We compare three strategies for item selection:

- random selection in the interactive training set;
- mean, which selects the item for which the probability of the FM model is closest to 0.5;
- and variance, which selects the item for which the variance of prediction is the highest.

[1]https://sharedtask.duolingo.com/2018.html

B. Models

The main baselines are the libFM[2] implementation of Gibbs sampling (MCMC) for training FMs [2], and variational Bayes factorization machines (VBFM) and their online counterpart (OVBFM) from [20]. OVBFM provides stochastic online training compared to full batch training in VBFM, however the OVBFM implementation from the authors is not expected to work on classification datasets.

We didn't include any deep model in our benchmarks, as they are often beaten by simpler models in collaborative filtering [17]. Still, we tried DeepFM on Movie100k and got a higher RMSE of 0.93 than the other baselines, although we couldn't perform cross-validation due to time constraints.

C. Framework

For the regression and classification tasks, 20% of the training set was used for cross-validating the embedding size d of FMs. We found that among $\{2, 5, 10\}$, $d = 5$ got the best results on the validation set. In all experiments we increased our ELBO estimate using the Adam optimizer [27] over full batch for Movie100k and Movie1M, using learning rate $\gamma = 0.1$. We used $S = 1$ variational sample, except for the preference elicitation task where $S = 100$.

w_0, $\boldsymbol{w}$ and V embeddings were initialized using $\mathcal{N}(0, 1)$, precision parameters λ initialized with 0.02. To enforce positivity of the standard deviation, we can use absolute value $|\cdot|$ or softplus : $x \mapsto \log(1 + \exp(x))$. We chose the latter.

For all models considered, all hyperparameters, notably α, were learned during training. For VFM, the stopping rule for validation was when the validation metric (RMSE or AUC) was worsening for 10 consecutive steps. For training or refit, it was when the ELBO estimate decreased during 4 consecutive steps. Our implementation, available online[3], is written in TensorFlow 1.15[4] using TensorFlow Distributions and TensorFlow Probability [28]. We also provide a simpler implementation in PyTorch using the torch.distributions package.

Using probabilistic programming, we can code the log-likelihood term in the ELBO as

```
outputs.log_prob(observed).mean()
```

using the same expression in the classification and the regression case.

VI. Discussion

The results are reported in Figure 3 and Table III for the regression and classification tasks, and in Figure 4 and Table IV for the preference elicitation task. For all models, considering the average of weights always leads to better performance than considering only the weights of the last epoch. Such an approach is called stochastic weights averaging [23], [24].

On regression tasks, all models have comparable performance. VBFM is the fastest to converge, thanks to their closed

[2]http://www.libfm.org

[3]https://github.com/jilljenn/vae/

[4]Good old times where TF had more identity than just Keras.

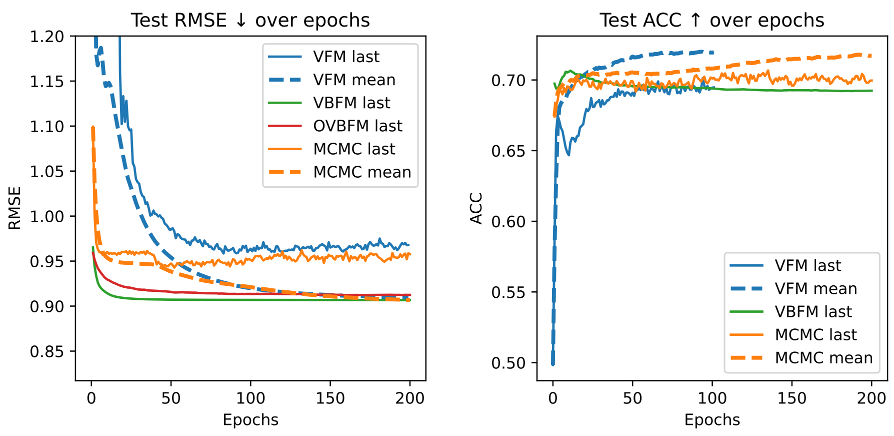

Fig. 3: Test metrics of all models on Movielens 100k datasets (regression task on the left, classification task on the right). On the right, VFM stopped early at 100 epochs.

		FM MCMC [2]	VFM	VBFM [20]	OVBFM
Movie100k	RMSE	**0.906**	**0.906**	0.907	0.912
Movie1M	RMSE	**0.840**	0.854	0.856	0.846
Movie100k-binary	ACC	0.717	**0.722**	0.692	–
	AUC	0.788	**0.791**		
	MAP	0.811	**0.813**		
Movie1M-binary	ACC	0.739	**0.746**	0.732	–
	AUC	0.809	**0.818**		
	MAP	0.840	**0.851**		
Duolingo	ACC	**0.848**	0.846	0.842	–
	AUC	**0.814**	0.809		
	MAP	**0.948**	0.946		

TABLE III: Results on all datasets. First rows are on the regression task, the latter ones on the classification task. The best results are in bold. VBFM and OVBFM is not suited for classification, still it is possible to run VBFM but just to observe accuracy.

form expressions of gradients. VFM may converge in fewer epochs if we use more variational samples, which induces an artificial reduction in variance but with slower training.

It is natural that VFM may be beaten by MCMC, as the latter does not assume an approximate posterior but samples from the real posterior. Still, on most classification tasks, VFM outperforms MCMC and VBFM in all metrics considered. It also converges faster. Indeed, there is no closed form expression of the gradient updates so VBFM speedup does not hold here.

To optimize the ELBO, we use Adam while VBFM use Robbins-Monro to decay their learning rate [20]. Indeed, learning rate decays can accelerate convergence in convex scenarios, as we observe it on regression but not on our classification experiments [22]–[25].

For the preference elicitation task, we see that most strategies for item selection have comparable performance. The variance strategy can result in higher scores with few items. Selecting items of which the probability of being rated is closest to 0.5 (mean strategy) for the first batch of 4 items does not have a good performance because the resulting items may be collinear and not bring diversity. For this initial batch, random selection or highest variance is more informative. When asking 8 items over 100, the mean strategy performs best, followed by variance. However later, most strategies have comparable performance, with mean still achieving the best metrics after 20 items.

VII. Future work

We showed that on large-scale datasets, especially classification datasets, variational training of FMs could result in faster and better training. Our training relies on batches of data, which can be used in online inference of FMs, for active learning applications such as preference elicitation. We showed the impact of selecting items of probability close to 0.5 or with high variance in the predictions in order to acquire information efficiently, still many other criteria from information geometry can be used, such as maximizing the trace or determinant of Fisher information. We plan to derive entropy-based strategies, that is, reducing the expected entropy at most at each item. We couldn't do it in this paper as it required to simulate too many possibilities to select the best possible items. However with

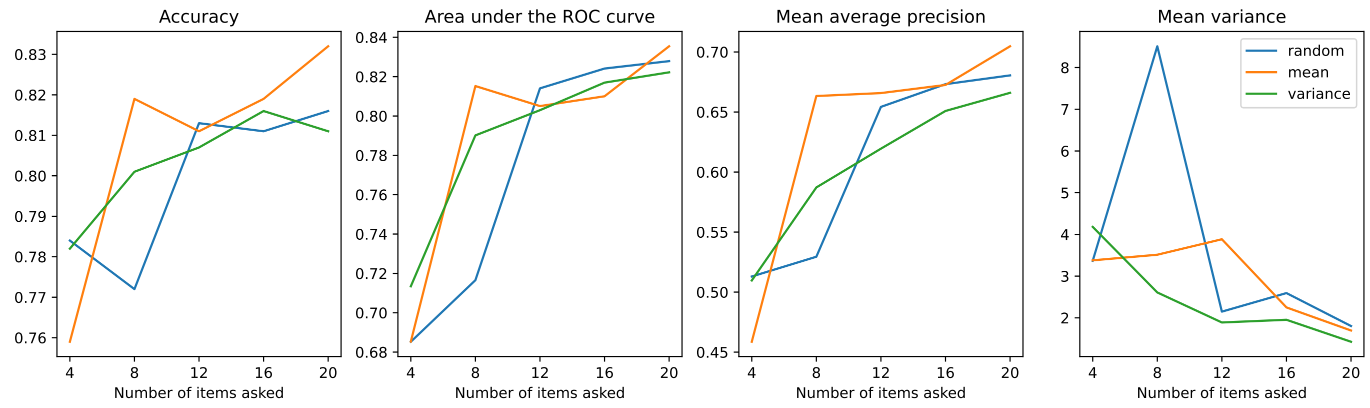

Fig. 4: Comparing strategies for preference elicitation on the Movie10k dataset.

Accuracy					
Items	4	8	12	16	20
Random	**0.784**	0.772	**0.813**	0.811	0.816
Mean	0.759	**0.819**	0.811	**0.819**	**0.832**
Variance	0.782	0.801	0.807	0.816	0.811

Area under the ROC curve					
Items	4	8	12	16	20
Random	0.685	0.717	**0.814**	**0.824**	0.828
Mean	0.685	**0.815**	0.805	0.810	**0.835**
Variance	**0.713**	0.790	0.803	0.817	0.822

Mean average precision					
Items	4	8	12	16	20
Random	**0.513**	0.529	0.654	**0.673**	0.680
Mean	0.459	**0.663**	**0.666**	**0.673**	**0.705**
Variance	0.510	0.587	0.619	0.651	0.666

Mean variance of predictions					
Items	4	8	12	16	20
Random	**3.368**	8.510	2.149	2.594	1.804
Mean	3.379	3.513	3.884	2.250	1.697
Variance	4.181	**2.609**	**1.889**	**1.954**	**1.427**

TABLE IV: Results on a validation set of different strategies for item selection in an interactive set, on the Movie10k dataset.

careful derivation of formulas, such updates can be computed efficiently.

In this paper, we saw the impact of stochastic weight averaging on performance. Other techniques can be tested, such as putting a stronger regularization on popular features in the dataset, just like ALS-WR (alternating least squares with weighted λ regularization) does in collaborative filtering [29].

There sure is a jungle of deep models for factorization but they contain even more hyper-parameters (number of layers, number of neurons) to calibrate. As the input data is sparse, those deep models may overfit. The interested reader may want to look at [18], [30], [31] for state-of-the-art benchmarks[5].

Berg et al. have shown that matrix completion can be modeled as message-passing on a graph [32]. The variational stochastic gradient updates in this paper can also be framed this way, as collaborative filtering can be seen as edge prediction in a bipartite graph of users and items. We plan to make more contributions in this direction, where side information such as movie posters [33] could be embedded in the graph and speed up preference elicitation.

We showed that a Gaussian approximation performs well, but it can be replaced with other, more complex families using normalizing flows or even stable diffusion [34], as long as the Jacobian is easy to compute [35].

[5]https://deepctr-doc.readthedocs.io/en/latest/Features.html

VIII. Conclusion

In this paper we showed how factorization machines could be trained easily by optimizing a simple variational objective. By recovering an approximation of the posterior distribution over the parameters of the model, one can identify which features need extra data points. Another advantage is that hyperparameter optimization is automatically done during training. This work opens the door to new techniques of active learning, or upper confidence bound algorithms from multi-armed bandits, because now reward functions can be expressed in function of the uncertainty of the model. We hope to see many applications in adaptive education and interactive recommender systems.

IX. Acknowledgements

We thank Akash Srivastava for friendly and enlightened discussions in Tokyo. This work was started while Jill-Jênn Vie was a researcher in Kyoto University funded by RIKEN AIP; and completed while the first two authors were visiting the third author in Kyoto U, funded by the OPALE associated team between Inria and Kyoto U. This work was supported by JST CREST Grant Number JPMJCR21D1.

References

[1] Y. Zhou, D. Wilkinson, R. Schreiber, and R. Pan, "Large-scale parallel collaborative filtering for the netflix prize," in *International Conference on Algorithmic Applications in Management*. Springer, 2008, pp. 337–348.

[2] S. Rendle, "Factorization machines with libfm," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 3, no. 3, pp. 57:1–57:22, 2012.

[3] N. Golbandi, Y. Koren, and R. Lempel, "Adaptive bootstrapping of recommender systems using decision trees," in *Proceedings of the fourth ACM international conference on Web search and data mining*, 2011, pp. 595–604.

[4] Y.-D. Kim and S. Choi, "Scalable variational bayesian matrix factorization with side information," in *Artificial Intelligence and Statistics*. PMLR, 2014, pp. 493–502.

[5] L. Cai, "High-dimensional exploratory item factor analysis by a metropolis–hastings robbins–monro algorithm," *Psychometrika*, vol. 75, no. 1, pp. 33–57, 2010.

[6] I. Porteous, A. U. Asuncion, and M. Welling, "Bayesian matrix factorization with side information and dirichlet process mixtures." in *AAAI*, 2010.

[7] C. Wang, Q. Liu, R.-z. Wu, E. Chen, C. Liu, X. Huang, and Z. Huang, "Confidence-aware matrix factorization for recommender systems." in *AAAI*, 2018.

[8] M. D. Hoffman, D. M. Blei, C. Wang, and J. Paisley, "Stochastic variational inference," *The Journal of Machine Learning Research*, vol. 14, no. 1, pp. 1303–1347, 2013.

[9] D. P. Kingma and M. Welling, "Stochastic gradient vb and the variational auto-encoder," in *Second International Conference on Learning Representations, ICLR*, 2014.

[10] J.-J. Vie and H. Kashima, "Knowledge tracing machines: Factorization machines for knowledge tracing," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 750–757.

[11] M. D. Reckase, "Multidimensional item response theory models," in *Multidimensional Item Response Theory*. Springer, 2009, pp. 79–112.

[12] M. Wu, R. L. Davis, B. W. Domingue, C. Piech, and N. Goodman, "Variational item response theory: Fast, accurate, and expressive." *International Educational Data Mining Society*, 2020.

[13] H. Guo, R. Tang, Y. Ye, Z. Li, and X. He, "Deepfm: a factorization-machine based neural network for ctr prediction," in *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, 2017, pp. 1725–1731.

[14] J.-J. Vie, "Deep factorization machines for knowledge tracing," in *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, 2018, pp. 370–373.

[15] M. Blondel, A. Fujino, and N. Ueda, "Convex factorization machines," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2015, pp. 19–35.

[16] M. Blondel, A. Fujino, N. Ueda, and M. Ishihata, "Higher-order factorization machines," in *Advances in Neural Information Processing Systems*, 2016, pp. 3351–3359.

[17] M. Ferrari Dacrema, P. Cremonesi, and D. Jannach, "Are we really making much progress? a worrying analysis of recent neural recommendation approaches," in *Proceedings of the 13th ACM conference on recommender systems*, 2019, pp. 101–109.

[18] J. Zhu, J. Liu, S. Yang, Q. Zhang, and X. He, "Barsctr: Open benchmarking for click-through rate prediction," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2021, pp. 2759–2769.

[19] C. E. Moody, "Variational factorization machines," 2016.

[20] A. Saha, R. Misra, A. Acharya, and B. Ravindran, "Scalable variational bayesian factorization machine," 2017.

[21] C. Doersch, "Tutorial on variational autoencoders," Tech. Rep., 2016.

[22] B. T. Polyak and A. B. Juditsky, "Acceleration of stochastic approximation by averaging," *SIAM journal on control and optimization*, vol. 30, no. 4, pp. 838–855, 1992.

[23] G. Neu and L. Rosasco, "Iterate averaging as regularization for stochastic gradient descent," in *Conference On Learning Theory*. PMLR, 2018, pp. 3222–3242.

[24] W. J. Maddox, P. Izmailov, T. Garipov, D. P. Vetrov, and A. G. Wilson, "A simple baseline for bayesian uncertainty in deep learning," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[25] P. Izmailov, D. Podoprikhin, T. Garipov, D. Vetrov, and A. G. Wilson, "Averaging weights leads to wider optima and better generalization," in *34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018*. Association For Uncertainty in Artificial Intelligence (AUAI), 2018, pp. 876–885.

[26] B. Settles, C. Brust, E. Gustafson, M. Hagiwara, and N. Madnani, "Second language acquisition modeling," in *Proceedings of the thirteenth workshop on innovative use of NLP for building educational applications*, 2018, pp. 56–65.

[27] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[28] J. V. Dillon, I. Langmore, D. Tran, E. Brevdo, S. Vasudevan, D. Moore, B. Patton, A. Alemi, M. Hoffman, and R. A. Saurous, "Tensorflow distributions," *arXiv preprint arXiv:1711.10604*, 2017.

[29] Y. Zhou, D. Wilkinson, R. Schreiber, and R. Pan, "Large-scale parallel collaborative filtering for the netflix prize," in *International conference on algorithmic applications in management*. Springer, 2008, pp. 337–348.

[30] W. Shen, "Deepctr: Easy-to-use,modular and extendible package of deep-learning based ctr models," https://github.com/shenweichen/deepctr, 2017.

[31] J. Zhu, K. Mao, Q. Dai, L. Su, R. Ma, J. Liu, G. Cai, Z. Dou, X. Xiao, and R. Zhang, "Bars: Towards open benchmarking for recommender systems," *arXiv preprint arXiv:2205.09626*, 2022.

[32] R. v. d. Berg, T. N. Kipf, and M. Welling, "Graph convolutional matrix completion," 2018, presented at the KDD 2018 Deep Learning Day.

[33] J.-J. Vie, F. Yger, R. Lahfa, B. Clement, K. Cocchi, T. Chalumeau, and H. Kashima, "Using posters to recommend anime and mangas in a cold-start scenario," in *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, vol. 3. IEEE, 2017, pp. 21–26.

[34] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 10 684–10 695.

[35] L. Dinh, J. Sohl-Dickstein, and S. Bengio, "Density estimation using real nvp," *arXiv preprint arXiv:1605.08803*, 2016.