Context-aware Preference Modeling with Factorization

Balázs Hidasi
balazs.hidasi@gravityrd.com

Gravity Research and Development Inc.
Mészáros utca 58/B.
1016 Budapest
Hungary

Budapest University of Technology and Economics
Magyar Tudósok krt. 2
1117 Budapest
Hungary

ABSTRACT

This work focuses on solving the context-aware implicit feedback based recommendation task with factorization and is heavily influenced by the practical considerations. I propose context-aware factorization algorithms that can efficiently work on implicit data. I generalize these algorithms and propose the General Factorization Framework (GFF) in which experimentation with novel preference models is possible. This practically useful, yet neglected feature results in models that are more appropriate for context-aware recommendations than the ones used by the state-of-the-art. I also propose a way to speed up and enhance scalability of the training process, that makes it viable to use the more accurate high factor models with reasonable training times.

Categories and Subject Descriptors

I.2.6 [[**Artificial Intelligence**]]: Learning - Parameter Learning

General Terms

Algorithms, Experimentation

Keywords

recommender systems; context-awareness; factorization; preference modeling; implicit feedback; scalability

1. INTRODUCTION

Recommender systems are more and more widely used in e-commerce and on multimedia sites. My work focuses on advanced algorithms that can be used in practical recommender system. One of the biggest distinction between practice and academic research is that the latter focuses on explicit feedback and the prediction of ratings, while the former uses implicit feedback and requires top N recommendations. Although there has been a shift towards this more practical setting in the last few years, the majority of research still focuses on ratings.

Implicit feedback is collected via monitoring the behaviour of users while they use a service (e.g. a web shop). User interaction is not required in order to get the feedback, therefore it is available in large quantity. This is of key importance in practical scenarios. Explicit feedback on the other hand is usually either not available or its amount is negligible compared to implicit feedback. The primary challenge of implicit feedback is that it does not explicitly encode user preferences. These preferences must be inferred from the interactions. The presence of an user action on an item (e.g. purchase) is considered to be a noisy sign of positive preference. However it is much harder to infer negative feedback as the absence of an event can be traced back to multiple causes, the most common being that the user does not know about the item. Although there are some ways to infer negative feedback in special cases, it is generally assumed to be missing and the absence of positive feedback is considered as a very week sign of negative preference.

Context-aware recommender systems (CARS) consider additional information (termed context) besides user–item interactions. Any information can be considered as context, however I argue that it is useful to distinguish event context from other types of data, such as item metadata, sociodemographic information or social network of the user. The common property of latter categories is that they are bound either to the item or to the user. Also, they are thoroughly examined in specific research topics, such as content based or hybrid recommenders. On the other hand, event context is associated with the interaction of the users and items and can not be bounded to either one. Typical examples are the time or the location of the event. The hypothesis of context-aware recommendations is that they can significantly improve recommendation accuracy, because (1) context related effects can be handled during training; (2) recommendation lists can be tailored according to the actual value of the context, which may influence the users' needs.

One of the most extensive data models for representing context-enhanced data is the Multidimensional Dataspace Model (MDM) [1] in which the dataspace is the Cartesian product of several dimensions, and each dimension is the Cartesian product of one or more attributes. Attributes are atomic and nominal and their value comes from a finite set of values. Almost all practically used context-enhanced data can be expressed in a more simple dataspace model,

The work leading to these results has received partial funding from the European Union's Seventh Frame work Programme (FP7/2007-2013) under CrowdRec Grant Agreement n° 610594

RecSys'15, September 16–20, 2015, Vienna, Austria.
Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM 978-1-4503-3692-5/15/09 ...$15.00.
DOI: http://dx.doi.org/10.1145/2792838.2796543 .

where each dimension consist of exactly one attribute. I refer to this dataspace model as single attribute MDM or SA-MDM. Note that if data is representable in SA-MDM it is also representable in a tensor. The SA-MDM representation is powerful enough for commonly used context dimensions, such as time or location[1].

Latent feature based collaborative filtering methods—such as matrix factorization—have gained popularity in the last decade due to their high accuracy and good scalability. My work focuses on factorization methods for implicit feedback that also incorporate context dimensions.

2. PREFERENCE MODELS & CONTEXT

In my early work I proposed a context-aware tensor factorization algorithm—iTALS (implicit Tensor Alternating Least Squares) [6]—for the implicit feedback problem. The algorithm works with SA-MDM. The presence of an event (i.e. a combination of attributes is in the training data) is considered to be strong positive preference, while the absence of an event is considered to be weak negative preference. The method uses pointwise ranking by optimizing for weighted root mean squared error (wRMSE). The value of the target is 1 for positive and 0 for negative feedback; the weight for positive feedback is much higher than for negative. Interaction data is inherently sparse and the usage of additional context dimension makes it even sparser; thus majority of the values in the tensor are zeros. However, contrary to explicit problems, there are no missing values in the tensor that makes the direct application of explicit algorithms inefficient. In iTALS, each dimension is assigned with a feature matrix that contains feature vectors of K length for all of the possible attribute values in that dimension. For the prediction of preferences iTALS uses the N-way model, i.e. the "dot product" of one feature vector from every dimension, corresponding to the user–item–context(s) configuration on which the prediction is requested.

The iTALSx algorithm [4][5] is a variation of iTALS. The difference is in the prediction model for which iTALSx uses a variant of the pairwise model, i.e. the sum of dot products between pairs of feature vectors. iTALSx considers only the user–item, user–context(s) and item–context(s) pairs[2]. The change in the model affects the optimization procedure.

Generalizing the idea of factorization, preference models—i.e. the expression with which the preference (or rating) is approximated—can be considered as the sum of various interactions. An interaction type is the dot product of feature vectors from selected dimensions, one vector per dimension (e.g. the user and the item feature vector for the given user and item). With only the user and item dimensions, there is only one possible interaction type: the user–item interaction (UI). This results in one possible factorization model. Adding one context dimension (S) gives the following new interaction types. USI is a reweighted user–item interaction where the weight is dependent on the value of the context. US and IS are context-dependent user and biases. The number of different models with 3 dimensions is 15. Adding another context (Q) further increases the number of interaction types. Besides UQI, UQ and IQ, there is $USQI$ (the user–item interaction whose reweighting depends on both context dimensions); USQ, ISQ and SQ in which there is some kind of interaction between context dimensions. The number of possible models with 4 dimensions is 2047.

However, not all of the interaction types are of equal importance. Generally items are recommended to users, therefore these two dimensions are more important. Also, users are the only entities which act and the target of these actions are the items. Context on the other hand is not a direct participant in the transactions, but may influence behaviour.

Despite of the large number of models, only two of them are used widely in the literature: the N-way and the full pairwise interaction model. Both of these models are symmetric, thus all dimensions are considered to be equal.

The choice of the model affects the optimization, but it is ineffective to implement a new version of an algorithm for every one of them. Therefore I created the General Factorization Framework (GFF) [7], which is a single flexible algorithm that takes the preference model as an input and does the computations accordingly. GFF allows its user to easily experiment with various linear models on any context-aware recommendation task. The following properties were important at the design of GFF.

1. No restriction on context[3]: GFF works on any context-aware recommendation problem independently of the number and the meaning of context dimensions.
2. Large preference model class: the only restriction on the preference model is that it must be linear in the dimensions of the problem[4]. This restriction does not restrict the applicability to real-world problems.
3. Data type independence: the implicit case is in the focus, but explicit problems can be also addressed by simply changing the weighting scheme in the loss function.
4. Flexibility: the weighting scheme of GFF is very flexible, enabling to incorporate extra knowledge through the weights such time decay, dwell time dependent weighting, missing not at random hypotheses and more.
5. Scalability: GFF scales well both in terms of the number of interactions in the training set and in the number of features. This makes it applicable in real life recommender systems. (See Section 3.)

Along with the users (U) and items (I), I used two context dimensions—seasonality (S) and sequentiality (Q)—that are available with every implicit datasets as long as the timestamp of the events is recorded. Their availability and usefulness makes these dimensions suitable for the experiment.

Seasonality: Many application areas of recommender systems exhibit the seasonality effect, because periodicity can be observed in many human activities. Thus seasonal data is an obvious choice for context [9]. First, the length of the season has to be defined. No repetitions are expected in the aggregated behavior of users within a season, and similar aggregated behaviour is expected at the same time offset in different seasons. Next, *time bands* need to be created within the seasons that are the context-states. Time bands specify the time resolution of a season. The length of time

[1]Even context dimensions that contain more than one attribute can be represented in SA-MDM, but less effectively.

[2]Other methods, such as Factorization Machines [11] also use context–context pairs in their variation of the this model.

[3]The basic GFF builds on SA-MDM, but the algorithm also has an extension that is compatible with the full MDM.

[4]Meaning that a dimension can not directly interact with itself in the model.

bands can be equal or different. In the final step, events are assigned to time bands according to their time stamp.

Sequentiality: In some domains, like movies or music, users consume similar items. In other domains, like electronic gadgets or e-commerce in general, they avoid items similar to what they already consumed and look for complementary products. Sequential patterns can be observed on both domain types. Sequentiality was introduced in [6] and uses the previously consumed item of the user as a context for the actual item. This information helps in the characterizations of repetitiveness related usage patterns and sequential consumption behavior.

Besides the traditional N-way and pairwise models, I selected the following preference models for experimentation.

- **Interaction model ($\boldsymbol{UI + USI + UQI}$):** The model is the composite of the base behavior of the users (UI) and the context-influenced modification of this behavior (USI and UQI). This model assumes that the preferences of the users can be divided into context independent and dependent parts.
- **Context interaction model ($\boldsymbol{USI + UQI}$):** Preferences in this model are modeled by solely context dependent parts, i.e. it assumes that user–item interactions strongly depend on the context and this dependency affects the whole interaction.
- **Reduced pairwise model ($\boldsymbol{UI + US + IS + UQ + IQ}$):** This model is a minor variation of the traditional pairwise model with the exclusion of the interaction between context dimensions (SQ). The interaction with context is done separately by users and items.
- **User bias model ($\boldsymbol{UI + US + UQ}$):** Here it is assumed that only the user interacts with the other dimensions. This results in a model where the user–item relation is supported by context dependent user biases. Note that during recommendation the user biases are constant, thus do not affect the ranking. But they filter out some context related noise during training.
- **Item bias model ($\boldsymbol{UI + IS + IQ}$):** This model assumes that the effect of context can described by context dependent item biases (e.g. items are popular under certain conditions). The item biases affect the ranking as well as filter context related noise.
- **A complex model ($\boldsymbol{UI + US + IS + UQ + IQ + USI + UQI}$):** This model is the composite of the reduced pairwise and the interaction model. It can be also treated as a reduced 3-way interaction model from which the context-context interactions are omitted.

2.1 Evaluation of models

I used five genuine implicit feedback data sets to evaluate the models in GFF. Three of them are public (LastFM 1K, [2]; TV1, TV2, [3]), the other two are proprietary (Grocery, VoD). The properties of the data sets are summarized in Table 1. The train–test splits are time-based: the first event in the test set falls chronologically after the last event of the training set. Artists were used as items in LastFM.

Table 1: Main properties of the data sets

Dataset	Domain	Training set			Test set	
		#Users	**#Items**	**#Events**	**#Events**	**Length**
Grocery	E-grocery	24947	16883	6238269	56449	1 month
TV1	IPTV	70771	773	544947	12296	1 week
TV2	IPTV	449684	3398	2528215	21866	1 day
LastFM	Music	992	174091	18908597	17941	1 day
VoD	IPTV/VoD	480016	46745	22515406	1084297	1 day

The primary evaluation metric is recall@20. The reason for using recall@N is threefold: (1) in live recommender systems recall correlates well with click-through rate (CTR), an important metric for recommendation success. (2) Recall@20 is a good proxy of estimating recommendation accuracy offline for real-world applications[6][10]. (3) Recall is event based, while ranking based metrics like MAP and NDCG are query based. The inclusion of context changes the query set of the test data, therefore the comparison by query based metrics is unfair.

The hyperparameters of the algorithms, such as regularization coefficients were optimized on a part of the training data (validation set). Then the algorithm was trained on the whole training data (including the validation set) and recall was measured on the test set. The number of epochs was set to 10, because all methods converge in at most 10 epochs. The number of features was set to $K = 80$ that is a good trade-off between accuracy and training time in practice.

Table 2: Recall@20 values for models within GFF. Grey: traditional models. Bold: best results.

Model	Grocery	TV1	TV2	LastFM	VoD
$USI + UQI$	0.1504	**0.1551**	0.2916	0.1984	0.1493
$UI + USI + UQI$	**0.1669**	0.1482	**0.3027**	**0.2142**	**0.1509**
$USQI$	0.1390	0.1315	0.2009	0.1906	0.1268
$UI+US+IS+UQ+IQ$	0.1390	0.1352	0.2388	0.1884	0.0569
$UI + US + UQ$	0.1619	0.0903	0.1399	0.1993	0.0335
$UI + IS + IQ$	0.1364	0.1266	0.2819	0.1871	0.1084
$UI+US+IS+UQ+IQ + SQ$	0.1388	0.1344	0.2323	0.1873	0.0497
$UI+US+IS+UQ+IQ + USI + UQI$	0.1389	0.1352	0.2427	0.1866	0.0558

Table 2 shows the accuracy of two traditional models and six novel models. There exists a novel model with all five datasets that performs better than both traditional models. 4 out of 5 cases the interaction model ($UI + USI + UQI$) is the best and it is the second best in the remaining one case. Thus this model is not only intuitively sound but also performs well that underpins its assumptions.

3. SCALABILITY IMPROVEMENT

GFF, iTALS and iTALSx use Alternating Least Squares (ALS) during training. In ALS, feature matrices are computed in an alternating fashion and all but the currently computed matrix are fixed. The efficient usage of ALS with implicit problems is not straightforward, although it is possible with the smart separation of computations. The computation of one feature matrix can be divided into three phases[6][5]: (1) computing common statistics[5] that are the same for all feature vectors; (2) feature vector specific update of the statistics using the training events; (3) solving a $K \times K$ sized system of linear equations per feature vector.

This way, the complexity of one epoch (computing each feature matrix once) is $O(N_D|O|N^+K^2+\sum_{i=1}^{N_D} S_i K^3)$, where N^+, K, N_D, S_i are the number of events, features, dimensions, different values of the attribute in the i^{th} dimension and $|O|$ is the complexity of the preference model. The method scales linearly with the number of events, which is very important in practice. It scales cubically with the number or features, but since $N_D|O|N^+ \gg \sum_{i=1}^{N_D} S_i$ the first

[5]A $K \times K$ sized matrix and a vector of K length.

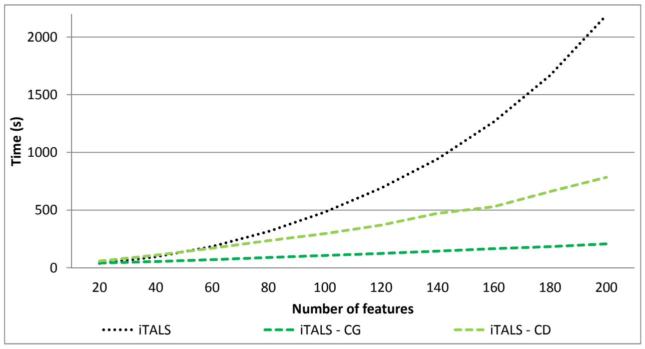

Figure 1: Time of one epoch of ALS, CD & CG with iTALS w.r.t. different number of features, using one CPU core

term is dominant in the range of practically used feature numbers an thus it scales quadratically with K in practice. This scaling property is fine for smaller K values but the training takes a lot of time for high factor models. For practical applicability, the training time of the algorithms is a key aspect. Faster training allows to (1) capture a more recent state of the system modeled; (2) retrain the models more frequently; (3) apply trade-off between running times and accuracy by using more features or running more epochs.

I created two approximate solutions that scale better in the number of features than ALS, but achieve similar recommendation accuracy[8]. The first uses Coordinate Descent (CD), an ALS variant in which each feature is computed separately while the others are fixed. The application of CD to this problem requires the efficient handling of the large amounts of "missing" feedback. CD does not approximate the ALS solution and has a complexity of $O(N_D|O|N_I N^+ K + \sum_{i=1}^{N_D} S_i K^2 N_I + N_D K^3)$ (linear in K in practice for the lower range of practical K values), where N_I is the number of inner iterations. The second method uses Conjugate Gradient (CG). CG is a fast way to approximate the solution of a system of linear equations with a symmetric coefficient matrix. The efficiency of CG depends on the efficiency of the matrix–vector multiplication between the coefficient matrix and a vector. In my methods the feature vector specific update is done by adding a dyadic sum to the common statistics. This allows the multiplication to be done in $K^2 + N_j^+ K$ time for the j^{th} feature vector. The complexity of one epoch can be reduced to $O(N_D|O|N_I N^+ K + \sum_{i=1}^{N_D} S_i K^2 N_I)$, that is linear in the number of features for the range of practical K values. CG approximates the ALS solution and yields the exact solution if the number of inner iterations equals to K.

Experimentation showed that ALS-CD and ALS-CG performs similarly to ALS in terms of recommendation accuracy. ALS-CD was found to be unstable with N-way components in the model if the number of features is high and one of the interacting dimensions is small. The speed up achieved by ALS-CG and ALS-CD is significant: for the commonly used $K = 80$ the speed up to ALS is ~ 3.5 and ~ 1.3 respectively; for $K = 200$ it increases to ~ 10.6 and ~ 2.9. Figure 1 shows the scaling of ALS, ALS-CG and ALS-CD with the number of features.

Overall, ALS-CG seems to be the better of the two approximation methods, due to its stability, speed, better approximation of ALS and other properties.

4. CONCLUSION & FUTURE WORK

My research is heavily influenced by practical considerations. It focuses on the implicit feedback problem and tackles it using context-awareness and factorization. The iTALS and iTALSx algorithms solve this task efficiently. These algorithms use different preference models and are better for different datasets. The preference modeling has been unjustly neglected in recommender related research. Therefore I created GFF, a single flexible algorithm that takes the preference model as an input and computes the latent feature matrices accordingly. Novel models were examined for a 4 dimensional context-aware problem using GFF. Novel models outperformed traditional ones and the *interaction model* performs well generally. Scalability is another important aspect in practice, therefore I proposed two solutions that enable the usage of high factor models for the implicit context-aware task. The ALS-CG method scales linearly in the number of features in practice that results in significant speed up compared to the basic ALS.

In future research I aim to add an automatic model learning feature to GFF. While the flexibility of GFF is a great asset for finding new models, the users of the framework may be overwhelmed by the possible preference models, especially with novel context dimensions. Although the intuitively sound interaction model is a good starting point, it may not be the best model for all context-aware problems. Therefore it would be useful if GFF could propose a good model for any context-aware recommendation problems.

5. REFERENCES

[1] Adomavicius, G., Sankaranarayanan, R., Sen, S., and Tuzhilin, A. Incorporating contextual information in recommender systems using a multidimensional approach. *ACM Trans. Inf. Syst. 23*, 1 (2005), 103–145.

[2] Celma, O. *Music Recommendation and Discovery in the Long Tail*. Springer, 2010.

[3] Cremonesi, P., and Turrin, R. Analysis of cold-start recommendations in IPTV systems. In *Recsys'09: ACM Conf. on Recommender Systems* (2009).

[4] Hidasi, B. Technical report on iTALSx. Tech. Report Series 2012-2, Gravity R&D Inc., 2012.

[5] Hidasi, B. Factorization models for context-aware recommendations. *Infocommunications Journal VI*, 4 (2014), 27–34.

[6] Hidasi, B., and Tikk, D. Fast ALS-based tensor factorization for context-aware recommendation from implicit feedback. In *ECML-PKDD'12, Part II*, no. 7524 in LNCS. Springer, 2012, 67–82.

[7] Hidasi, B., and Tikk, D. General factorization framework for context-aware recommendations. *Data Mining and Knowledge Discovery* (2015), 1–30.

[8] Hidasi, B., and Tikk, D. Speeding up ALS learning via approximate methods for context-aware recommendations. *Knowledge and Information Systems* (2015), 1–25.

[9] Liu, N. N., Zhao, B. C. M., and Yang, Q. Adapting neighborhood and matrix factorization models for context aware recommendation. In *CAMRa'10: Workshop on Context-Aware Movie Recommendation* (2010), 7–13.

[10] Liu, Q., Chen, T., Cai, J., and Yu, D. Enlister: Baidu's recommender system for the biggest Chinese Q&A website. In *RecSys-12: Proc. of the 6th ACM Conf. on Recommender Systems* (2012), 285–288.

[11] Rendle, S. Factorization machines with libFM. *ACM Transactions on Intelligent Systems and Technology (TIST) 3*, 3 (2012), 57.