Categorical Factorization Machines for Personalized Recommendation

Stanislas Bucaille
Research project under the supervision of Dr.Bonald

***Abstract*—In this paper, we are going to study a cutting-edge machine learning technique used for personalized recommendation: Factorization Machines (FMs). After defining what we imply by "recommendation" and "personalization", we will see how Factorization Machines can surpass Collaborative Filtering and Content-Based methods in providing accurate understanding and predictions of consumer's behavior using personalized variables. We will go even further and propose a modified implementation of FMs – Categorical Factorization Machines (C-FMs) – to show that there exist hidden correlations between these variables. Finally, by combining the collaborative and the content-based approaches, we will see that FMs offer a solution to the cold-start problem.**

***Keywords*—recommendation, factorization machine, cold-start problem.**

I. Introduction

The rise of data science in our society has enabled businesses to exploit consumer data to boost profitability. Understanding and predicting consumer behavior has become essential for these businesses to increase their sales by providing personnalized recommendations. Therefore, in the last decade, the field of recommender systems has gained significant importance and now holds a central position in many business strategies.

This work has been conducted on movie recommendation using the *ml-100k Movie Lens* dataset. The results of this paper apply to most recommendation problems.

II. Personalized Recommendation

Before getting into technicalities, it is important to keep in mind the big picture: when given a product, such as movies, how can we provide the best recommendations possible to users? A recommender system is said to be efficient when users are satisfied by the product they are being recommended. Thus, it feels natural to assume that the more personalized the recommendation, the best. But what do we mean by personalized recommendation?

In this paper, we define a recommender system as an algorithm capable of predicting users' tastes and preferences based on historical data. These historical data are commonly referred to as feedbacks, and can be implicit or explicit. For example, the fact that a user watches a certain movie is an implicit feedback because it does not say anything directly about the user's taste. On the other hand, a rating – on a scale of 1 to 5 – is an explicit feedback that clearly measures the user's satisfaction toward a given movie.

However, recommender systems are not always personalized. We say that a system is personalized when it is altered by personal variables. Unlike historical data, personal variables are static elements that characterize users and items. In the Movie Lens dataset used for this paper, we are given personalized variables for users – sexe, age, occupation – and for movies – genre.

In the next section, we see how both historical data and personal data can be used to provide recommendations.

III. Recommender systems

Recommender systems filter out items that a user might like and offer a list of suggestions based on rating predictions. These information filtering systems rely on the assumption that similar users have similar tastes. Thus, the problem lies in the way we define similarities between users and/or items. There exist different methods to computed these similarities.

In this section, we will study two different approaches: Collaborative Filtering and Content-Based.

A. Collaborative Filtering

Within recommender systems, collaborative filtering is the most well-known technique.

> *Collaborative filtering assumes that users who have shared common taste for similar movies in the past, are similar and will thereby continue to share common tastes in the future.*

In practice, methods of collaborative filtering use ratings made in the past to make their predictions. To get a better sense of it, let us look at Latent Matrix Factorization, one of the most popular method of collaborative filtering.

Latent Matrix Factorization model equation:

Given a user i and an item j represented by their embedding (or latent) vectors u_i and v_j, the predicted rating associated to this couple is defined as,

$$\hat{r}_{ij} = \langle u_i, v_j \rangle = \sum_{k=1}^{K} u_{ik} v_{jk} \tag{1}$$

where K is the dimension of the latent vectors.

As we can see, in the Matrix Factorization model, the rating prediction is computed by the interaction between the user and the item. This interaction accounts for their similarity.

In order to train this model, we decided to use a Triplet Loss because it is very well suited for ranking tasks:

$$\text{Loss} = \sum_{(u_i, v_p, v_n) \in T} \max(\langle u_i, v_p \rangle - \langle u_i, v_n \rangle + \alpha, 0) \quad (2)$$

where u_i represents a user, v_p a positive item and v_n a negative item.

As we can see, training the model on a Triplet Loss forces the recommender system to give a higher importance to the user's interactions with positive items – rated above 3 – than to that with negative items – not seen or rated under 3.

We can say that collaborative filtering is a *memory-based* method because it is trained using known ratings: historical data.

Based solely on the interaction between users and the products at stake, this method considers users for the type of consumers they are without trying to understand what makes them unique. Therefore, although this collaborative filtering method has proven to be extremely efficient for businesses to boost their sales, it fails to provide a personalized recommendation.

Furthermore, when facing a new user, this algorithm is unable to provide any recommendation because it requires non-existent historical data. This is was we call the cold-start problem.

The next method will offer a solution to this problem.

B. Content-Based method

The second approach – content-based – is based on users' and items' personal variables.

A content-based method assumes that users who share common personal variables are likely to share similar tastes, at any time.

An easy way to construct a recommender system using a content-based method is, for example, to group users based on their personal variables. The Movie Lens dataset provides a user table with users' age, sexe and occupation. Thus, we can use the K-Means algorithm on this dataset to create clusters of users. Given a user u in a cluster C and an item i, we then compute the average rating of the users in C on item i to make recommendations for user u.

Thus, unlike *collaborative filtering*, the *content-based* method is entirely personalized as it focuses sole on the similarities between users' personal variables.

Furthermore, the content-based method also solves the previously mentioned cold-start problem. When applying this content-based method on the Movie Lens data set, with K = 10, we get:

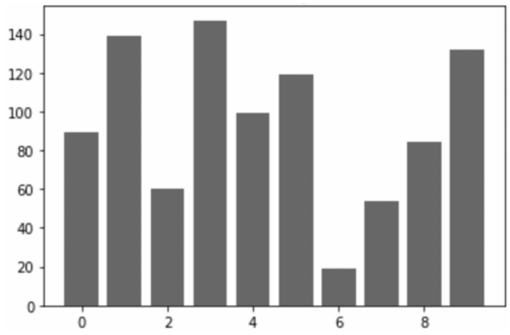

FIGURE I: NUMBER OF USERS PER CLUSTERS

In the graph above, we can see that users are nicely distributed in the different clusters. Thus, when a new user comes in, he will naturally fall in one of the clusters, which will enable us to make recommendations.

However, there are two major problems with this type of method. First, in order to expect good performances from a content-based method, one must have access to a feature rich dataset, which can be very difficult to acquire. Second, because recommendation is a very sparse problem, there are lots of chances that there exist movies which have been rated by none of the users in your cluster. In such a case, the method fails to solve the cold-start problem.

We can note that there exists a variant where we run the K-Means algorithm on the items table provided by the Movie Lens dataset instead of the users table, but this version does not perform as well and systematically fails to support the cold-start problem.

C. Conclusion

We just looked at two types of recommender systems using different approaches on different data. By weighting their pros and cons, we can see that Collaborative Filtering and Content-Based methods are extremely complementary. Thus, this realization led researchers to create a hybrid algorithm combining both approaches: Factorization Machines.

IV. FACTORIZATION MACHINES

Introduced by Steffen Rendle in 2010, Factorization Machines (FMs) are state-of-the-art factorization models created based on the idea that there exist hidden correlations between variables of a given model. Indeed, FMs offer high order – complex – interactions between variables, where the parameter d defines the highest order interaction.

The general model equation for a Factorization Machine of degree d is defined as:

$$\begin{aligned} \hat{y}(x) := w_0 + \sum_{i=1}^{n} w_i x_i \\ + \sum_{l=2}^{d} \sum_{i_1=1}^{n} \cdots \sum_{i_l=i_{l-1}+1}^{n} \left(\prod_{j=1}^{l} x_{i_j} \right) \left(\sum_{f=1}^{k_l} \prod_{j=1}^{l} v_{i_j,f}^{(l)} \right) \end{aligned} \quad (3)$$

This cutting-edge factorization model is well suited for recommender systems because it combines the advantages

of the collaborative filtering and content-based approaches. Indeed, we will see that an FM model is trained using both the historical data – the ratings – and the personal variables of users and items.

In addition, on a more practical note, FMs also gained success from their ability to handle problems with huge sparsity. Indeed, FMs use factorized parameters to model the interactions between their variables which break the dependances in the interaction parameters.

In this paper, we decided to work with a second order FM (d=2) because the sparse setting of recommender systems was not rich enough to compute interactions of highest orders. The model equation for a Factorization Machine of degree d=2 is defined as:

$$\hat{y}(x) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle v_i, v_j \rangle x_i x_j \qquad (4)$$

where the parameters are $w_0 \in R, w \in R^n, V \in R^{n \times k}$

We can notice that this model generalizes the previously seen Latent Matrix Factorization model while incorporating personal variables. It computes the user to item interaction, as well as interactions between the personal variables.

Indeed, FM models are extremely powerful because they can easily incorporate extra features using a particular dataset format. As we can see below, in order to incorporate a new category – a personal variable – it suffices to turn this variable into one hot encoding vector then concatenate it to the current feature vectors.

FIGURE II. DATASET STRUCTURE

	User				Item				Sexe		Age			Occupation				Genre				Rating	
$x^{(1)}$	1	0	0	...	1	0	0	...	1	0	0	1	0	1	0	0	...	1	0	1	...	2	$y^{(1)}$
$x^{(2)}$	1	0	0	...	0	1	0	...	0	1	1	0	0	0	1	0	...	0	0	0	...	4	$y^{(2)}$
$x^{(3)}$	0	1	0	...	1	0	0	...	0	1	0	1	0	0	0	0	...	1	0	0	...	5	$y^{(3)}$
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
$x^{(N)}$	0	0	0	...	0	0	0	...	1	0	0	0	1	0	1	0	...	0	1	0	...	3	$y^{(N)}$

where $x^{(i)} = (x_1^{(i)}, ..., x_n^{(i)})$.

Looking at the very organized data structure, we can rewrite (4) in order to identify the interactions between the different categories.

$$\hat{y}(x) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{c,d \in C} \sum_{\substack{i \in I_c \\ j \in I_d}} \langle v_i, v_j \rangle x_i x_j \qquad (5)$$

where $(I_c)_{c \in C}$ is a partition of $I = \{1, ..., n\}$ and C is the set of categories of the vectors' features.

This is the most important point of the paper. As we have seen earlier, machine learning algorithms are not only trying to reproduce forms of human intelligence such as being able to tell a cat from a dog on an image; their real strength comes from their ability to detect and understand the hidden correlations – interactions – between the model's variables.

However, although these algorithms are able to take advantage of hidden correlations invisible to the naked eye, they are rarely capable of providing an explanation for their reasoning. On that thought, we decided to propose a slightly modified version of classical FMs models, named Categorical Factorization Machines (C-FMs) and defined by the equation:

$$\hat{y}(x) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{c,d \in C} \lambda_{cd} \sum_{\substack{i \in I_c \\ j \in I_d}} \langle v_i, v_j \rangle x_i x_j \qquad (6)$$

As you can see, the equation is similar to the previous one (5), except that we added a factor λ_{cd} for every couple of categories c and d. We call these factors categorical factors.

This new model helps solve the lack of transparency in machine learning. In fact, once the model is trained, we can easily retrieve the *categorical factors* that will attest the importance of the interactions between categories. When training the C-FM on the *ml-100k Movie Lens* dataset, we found the following ranking for the importances of interactions in the context of movie recommendations:

TABLE I. VARIABLES INTERACTIONS

Interaction	**Importance**
Item & User	4.29
Item & Occupation	0.93
Item & Sexe	0.91
Item & Age	0.53
User & Genre	0.50
...	...
User & Occupation	0.01

These importances represent the degree to which each interaction contributes to the recommendation. As expected, the item-to-user interaction is by far the most important one. Coming second and third, we found the item-to-occupation and item-to-sexe interactions, followed in fourth and fifth position by the item-to-age and user-to-genre interactions. Qualitatively, this result seems coherent because it is safe to assume that a user chooses its occupation based on certain preferences that also drive its interest in certain type of movies. Moreover, sexe and age naturally play an important role in the recommendation. Finally, the user-to-genre interaction high-ranked position reflects users' tendency to prefer certain genders of movies more than others.

The low importance of the last interaction can be interpreted as follows. The user-to-occupation interaction may not be relevant to the problem of recommendation, even if these two variables are closely correlated.

V. RESULTS

In previous sections, we introduced several models of recommender systems. In this section, we will present the performances of these models in a general setting as well as in the cold-start problem situation. For the general setting, the models are trained and tested using the train and test sets provided by the Movie Lens dataset. For the cold-start problem, we decided to remove the last ten users - IDs 932 to

942 - from the train set for training and tested their movies' rating predictions using the test set. Finally, we choose the ROC-AUC score for the scoring metric because it is well suited for ranking problems.

Serving as an indicator for the other models, we also introduced a baseline model. The baseline model predicts, for a given movie, the same rating for every user by taking the average of the previous ratings made on this movie.

TABLE II. GENERAL SETTING

Model	Parameters	Score (ROC AUC)
Baseline	Loss: None Optimizer: None	0.69
Content-based using K-Means (K=10)	Loss: None Optimizer: None	0.67
Collaborative filtering Latent Matrix Factorization	Loss: Triplet Loss Optimizer: Adam	0.93
FM	Loss: Hinge Optimizer: Adam	0.93
C-FM	Loss: Hinge Optimizer: Adam	0.92

TABLE III. COLD-START PROBLEM

Model	Parameters	Score (ROC AUC)
Baseline	Loss: None Optimizer: None	0.69
Content-based using K-Means (K=10)	Loss: None Optimizer: None	0.67
Collaborative filtering Latent Matrix Factorization	Loss: Triplet Loss Optimizer: Adam	0.68
FM	Loss: Hinge Optimizer: Adam	0.85
C-FM	Loss: Hinge Optimizer: AdaM	0.86

We can see that the Collaborative Filtering model – Latent Matrix Factorization – performs very well in the general setting, but fails to solve the cold-start problem. On the other hand, although the Content-Based model – used with K-Means – performs poorly in the general setting, its performance does not deprecate when facing the cold-start problem. Its low performance can be explained by the model's inability to rate certain movies for users in small clusters, due to the sparse setting of recommendation.

Furthermore, we can see that the FM model does indeed take advantage of these two approaches; it performs as well as the collaborative filtering model in the general setting and maintains a high-performance score when facing the cold-start problem. Likewise, the C-FM model performs just like the FM model. This comes to no surprise knowing that the two models are almost identical.

VI. CONCLUSION

In this paper, we showed how Factorization Machines act as powerful techniques for personalized recommendation. As a hybrid algorithm between Collaborative Filtering and Content-Based methods, it enables better performance and solves the cold-start problem using hidden correlations between personal variables.

In addition, we offered a modified version of the algorithm - Categorical Factorization Machines - to underline the importance of variables' interactions. More specifically, this new algorithm focuses on the interaction between categories of variables through categorical factors that measure the importance of these interactions. This latter point is particularly welcomed nowadays because it gives us an insight on the algorithm's thinking process, when most deep learning methods lack this level of transparency.

REFERENCES

[1] S. Rendle, "Factorization Machines", Osaka University, 2010

[2] Nick P., "Factorization Machines for Recommendation Systems", Stream, 2016

[3] S. Narkhede, "Understanding AUC-ROC Curve", Towards Data Science, 2018

[4] A. Robicquet, "It shouldn't take half an hour to choose a movie", Medium, Crossing Mind, 2018

[5] E. Contal, "What are the differences between recommendation and personalization?", Medium, Crossing Mind, 2019

[6] T. Bonald, "Triplet Loss for Recommender Systems with Implicit Feedback", Télécom Paris, 2020