Learning Hierarchical Feature Influence for Recommendation by Recursive Regularization

Jie Yang*, Zhu Sun†, Alessandro Bozzon*, Jie Zhang†
*Delft University of Technology, †Nanyang Technological University
*{J.Yang-3, A.Bozzon}@tudelft.nl, †{SunZhu, ZhangJ}@ntu.edu.sg

ABSTRACT

Existing feature-based recommendation methods incorporate auxiliary features about users and/or items to address data sparsity and cold start issues. They mainly consider features that are organized in a flat structure, where features are independent and in a same level. However, auxiliary features are often organized in rich knowledge structures (e.g. hierarchy) to describe their relationships. In this paper, we propose a novel matrix factorization framework with *recursive regularization* – ReMF, which jointly models and learns the influence of hierarchically-organized features on user-item interactions, thus to improve recommendation accuracy. It also provides characterization of how different features in the hierarchy co-influence the modeling of user-item interactions. Empirical results on real-world data sets demonstrate that ReMF consistently outperforms state-of-the-art feature-based recommendation methods.

1. INTRODUCTION

Recommender systems aim to model user preferences towards items, and actively recommend relevant items to users. To address the *data sparsity* and *cold start* problems [23], feature-based recommendation methods, such as collective matrix factorization (CMF) [25], SVDFeature [4], and factorization machine (FM) [21], quickly gain prominence in recommender systems. They enable the integration of auxiliary features about users (e.g. gender, age) and items (e.g. category, content) with historical user-item interactions (e.g. ratings) to generate more accurate recommendations [24].

While commonly arranged in a flat structure (e.g. user gender, age), auxiliary features can be organized in a "feature scheme", i.e. a set of features that includes relationships between those features. Hierarchies are a natural yet powerful structure to human knowledge, and they provide a machine- and human- readable description of a set of features and their relationships. Typical examples of feature hierarchies include category hierarchy of on-line products (e.g. Amazon web store [19]), topic hierarchy of articles (e.g. Wikipedia [10]), genre hierarchy of music (e.g. Yahoo! Music), etc.. The benefits brought by the explicit modeling of feature relationships through hierarchies have been investigated in a broad spectrum of disciplines, from machine learning [12, 11] to natural language processing [10]. How to effectively exploit feature hierarchies in recommender systems is still an open research question. We here provide a running example to illustrate how hierarchical feature structure can provide better recommendations.

RecSys '16, September 15-19, 2016, Boston , MA, USA

DOI: http://dx.doi.org/10.1145/2959100.2959159

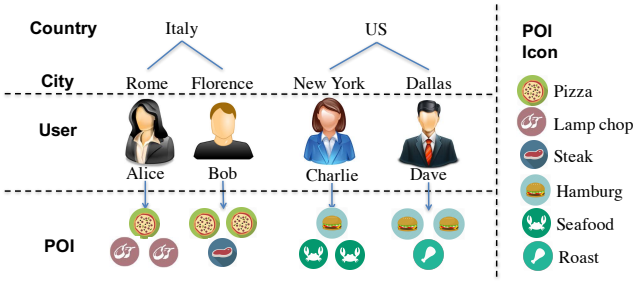


Figure 1: POI recommendation with auxiliary features hierarchy.

Running Example. Consider a point of interest (POI) recommender system [30], where the goal is to recommend a real-world POI (e.g. a restaurant) to a user. User preferences can be influenced by geo-cultural factors: for instance, the country of residence might have an influence on user preferences for restaurants' cuisine (intuitively, an Italian and an American might have different culinary tastes). Likewise, fellow countrymen might show different preferences according to their city of residence (arguably, citizens from Boston and San Diego can show incredible different culinary preferences). This scenario is represented in Figure 1: users are described by auxiliary features that characterize their countries and cities of residence. These features are organized in a hierarchy, where cities are related to countries by a *locatedIn* relationship.

We consider the historical POI interactions in a given city of four users in Figure 1: Alice and Bob, from Rome and Florence (Italy); Charlie and Dave from Boston and San Diego (US). Italian users (Alice and Bob) both show preferences towards Pizza, thus suggesting a "national imprint" on their preferences. However, the two users are differently influenced by their country of residence, as Alice's preference for pizza is weaker than that of Bob: Alice checks in more at Lamp chop restaurants, while Bob checks in more at Pizza restaurants. A similar observation holds also for Charlie and Dave: the influence of US is weaker than that of Boston on Charlie, while stronger than that of San Diego on Dave.

The example highlights how related features (a country and its cities, linked by the *locatedIn* relationship) can co-influence user preferences, although the strength of the co-influence varies across relationship instances (e.g. Italy-Rome, Italy-Florence). This observation suggests the need for feature relationships (e.g. the *locatedIn* relationship) to be properly considered in recommendation methods. This co-influence could be known as a priori, but it is often best learnt from historical user-item interaction data.

Existing feature-based methods, e.g. SVDFeature [5], CMF [25] and FM [21], ignore the useful information provided by feature relationships, imposing a conversion step that transforms a hierarchical structure into a flat one. To fully exploit feature hierarchies, the main challenge is to model the co-influence of features on user-item interactions, determined by both the feature relationships in the hierarchical structure and the historical user-item interaction data.

Original Contribution. We propose a novel approach that *models* the co-influence of hierarchically-organized features on user-item interactions, and *learns* the strength of such co-influence from historical user-item interaction data, to improve recommendation performance. We first define the influence of an individual feature as regularization on latent factors, then combine the regularization of individual features by weighting them recursively over the hierarchy, from root to leaves, according to their organization. The regularization of the feature hierarchy, named *recursive regularization*, is expressed as a regularization function parameterized by the weights associated to each feature. We then propose a novel recommendation framework `ReMF`, that integrates recursive regularization into the matrix factorization model to better learn latent factors. By learning the values of weights of each feature from the historical user-item interaction data, `ReMF` characterizes the influence of different features in a hierarchy on user-item interactions. We demonstrate the effectiveness of `ReMF` with an extensive validation performed on two recommendation scenarios, namely POI and product recommendation, and on multiple real-world data sets. Empirical results show that `ReMF` outperforms state-of-the-art approaches, scoring average improvements of 7.20% (MAE), 15.07% (RMSE) and 9.86% (AUC).

2. RELATED WORK

Incorporating auxiliary features into recommendation, i.e. feature-based recommendation [23, 24], has become a popular and effective approach to address the *data sparsity* and *cold start* problems. A wide range of features has been explored, including user gender and age [1, 6], item category [13] and content [20, 7].

Many feature-based recommendation methods consider only features with a flat structure. For example, Singh et al. [25, 17] propose the collective matrix factorization (CMF) method, which factorizes the user-item rating and user-feature matrices simultaneously, to improve the recommendation performance. Chen et al. [4] devise a machine learning tookit, named SVDFeature. The basic idea is that a user's (an item's) latent factor is influenced by those of her (its) features. Rendle et al. [21, 22] design factorization machines (FM) that combines the advantages of Support Vector Machines with factorization model. However, all of these methods mentioned above cannot cope with hierarchical feature structure. Blending a feature hierarchy into these models requires converting the hierarchy into a flat structure, thus losing the information about feature relationships. To fully exploit a feature hierarchy, `ReMF` combines the distinct influence of different features on user-item interactions according to their structured relationships.

Some studies on *taxonomy*-aware recommendation incorporate hierarchy in recommendation. For example, Ziegler et al. [33] and Weng et al. [29] propose to model a user's taxonomy preferences as a flat vector, where each element corresponds to the user's preference over a taxonomy feature. The user's preference is modeled as the frequency the user rates items characterized by the feature. Albadvi et al. [2] propose a similar method, however it models each feature as a preference vector, where the elements are feature attributes (e.g. price, brand). All of these methods ignore feature relationships. Koenigstein et al. [13] design a new matrix factorization model for Yahoo! Music competition that incorporates the feature hierarchy of track album and artist. They predict user preferences by fusing item (e.g. track) latent factors with feature (e.g. album, artist) vectors. This idea is similar to SVDFeature [4]. Though feature relationships are considered, they cannot fully exploit a feature hierarchy as they simply add feature latent factors to item latent factors, without taking into account the dependent influence of hierarchically-organized features on user-item interactions.

Another related line of research focuses on integrating the structure *within* users/items in recommendation, e.g. social network [27, 18, 26], webpage network [16], tag network [32]. These methods usually regularize latent factors of users/items that are linked in the network, based on heuristic definitions of similarity between users/items. For instance, Ma et al. [18] propose SoReg that regularizes user latent factors based on cosine similarity of ratings between socially connected users. These methods consider the network *within* users/items, though can be applied in the case of feature hierarchy, e.g. by constructing implicit connections between users/items according to their feature relationships in the hierarchy. However, an essential difference between these methods and ours is that the influence of features on user-item interactions considered in these methods is usually hard-coded with manually defined similarity between users/items; on the contrary our proposed framework can automatically learn the co-influence of different features from the historical user-item interaction data. Recently Wang et al. [28] propose to model the implicit hierarchical structure *within* users and items based on user-item interactions. Our work differs from this one, in that we consider leveraging explicit auxiliary features to guide the learning of latent factors.

In summary, existing methods are incapable to model the co-influence of hierarchically-organized features on user-item interactions, thus restricting their applications in recommendation. In contrast, our framework can fully exploit an auxiliary feature hierarchy through the learning of hierarchical feature influence.

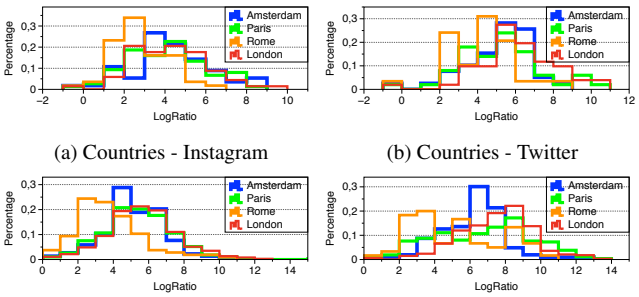
3. DATA ANALYSIS

This section demonstrates the need for recommendation methods able to account for the co-influence on user-item interactions of hierarchically-organized features.

Inspired by the running example, we show on 8 different data sets how: 1) users from the same country (named Country Visitors) are more similar in terms of POI preferences compared with users from different countries (named Foreign Visitors); and 2) users from the same city (named City Visitors) are more similar in terms of POI preferences compared to users from different cities *but the same country* (named Domestic Visitors).

Data Sets We collect data of Foursquare check-in's performed over 3 weeks in 4 European capital cities (Amsterdam, London, Paris, Rome) and published on 2 social media platforms (Twitter, Instagram). Table 1 shows the statistics about the 8 data sets. We consider users' residence *city*, *country* and *continent* as auxiliary information about users, as well as a root feature *residence location*. We use the method described in [3] to locate users' residence locations. For conciseness, we only analyze the co-influence of *country* and *city*. Overall we consider 121 countries and 2,873 cities.

Analysis Metrics. We denote all countries as $Con_1, \ldots, Con_s$; each country Con_s is the parent of all cities in it, i.e. $Con_s = parent(Cit_1, \ldots, Cit_t)$. Each user u_i from a city Cit_t and a country Con_s ($Con_s = parent(Cit_t)$), has a set of visited POIs, i.e. $POI(u_i) = \{poi_{i1}, poi_{i2}, \ldots\}$. Then we measure the similarity between the users u_i and u_k using Jaccard similarity, i.e. $Jar(u_i, u_k) = |POI(u_i) \cap POI(u_k)|/|POI(u_i) \cup POI(u_k)|$.



(a) Countries - Instagram (b) Countries - Twitter

(c) Cities - Instagram (d) Cities - Twitter

Figure 2: Distribution of the ratio between similarity (a, b) among countries and across countries, or (c, d) among cities and across cities, through the lens of (a, c) Instagram and (b, d) Twitter, including user visits to 4 European capital cities Amsterdam, Paris,Rome, and London (4 colors).

Table 1: Descriptive statistics of the data sets.

		Amsterdam	Rome	Paris	London
Inst.	#Users	4,318	4,081	11,345	12,719
	#POIs	5,768	7,878	14,849	12,892
	#Check-in's	28,142	26,714	80,553	66,092
	Sparsity	99.89%	99.92%	99.95%	99.96%
Twit.	#Users	1,599	1,369	6,521	9,305
	#POIs	3,816	4,876	16,046	14,117
	#Check-in's	8,670	8,727	43,541	48,852
	Sparsity	99.86%	99.87%	99.96%	99.96%

We define u_i's similarity with the other Country Visitors (City Visitors), and with all Foreign Visitors (Domestic Visitors) as

$$Sim(F, u_i^w) = \frac{1}{|F| - 1} \sum_{u_k \in F, u_i \neq u_k} Jar(u_i, u_k),$$

$$Sim(F, u_i^a) = \frac{1}{|parent(F)| - |F|} \sum_{u_k \in parent(F), u_k \notin F} Jar(u_i, u_k),$$

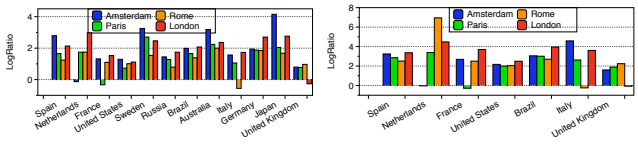
respectively, where F is the country (or city) u_i resides in, and $|F|$ is the number of users characterized by the feature F. For instance, in the case of $F = Con_s$, the similarity between u_i and the other Country Visitors, denoted by $Sim(Con_s, u_i^w)$, is the averaged similarity between u_i and each of the other Country Visitors; the similarity between u_i and Foreign Visitors, denoted by $Sim(Con_s, u_i^a)$, is the averaged similarity between u_i and each of the Foreign Visitors. The similarity between u_i and the other City Visitors and Domestic Visitors can be similarly calculated. Now we define the overall similarity within a country (city) F, and across the country (city) and other countries (cities) as $Sim(F^w) = [Sim(F, u_1^w), Sim(F, u_2^w), \ldots]$ and $Sim(F^a) = [Sim(F, u_1^a), Sim(F, u_2^a), \ldots]$, respectively. Then we compare the overall similarity within a country (city), and that across the country (city) and other countries (cities) by:

$$LogRatio(F) = Log_2\Big(\frac{1}{|F|} \sum_{u_i \in F} \frac{Sim(F, u_i^w)}{Sim(F, u_i^a)}\Big),$$

where $LogRatio(F) > 0$ indicates that the elements in $Sim(F^w)$ is larger than that in $Sim(F^a)$ on average, and $LogRatio(F)) < 0$ otherwise. We test the significance of the difference between $Sim(F^w)$ and $Sim(F^a)$ with a Paired t-test.

Observation 1: *Country Visitors are more similar with each other in terms of POI preferences than with Foreign Visitors.*

The distribution of $LogRatio(Con)$ for all countries is shown in Figures 2(a-b) for Instagram and Twitter, respectively. More than 95% of the countries observed in both Instagram and Twitter have $LogRatio(Con) > 0$. Paired t-test shows that 95.88% countries in Instagram and 99.36% in Twitter have $Sim(Con^w)$ significantly larger than $Sim(Con^a)$ (p-value < 0.01). We thus conclude that



(a) Countries - Instagram (b) Countries - Twitter

Figure 3: Ratio between similarity of users within a country and across the country and other countries, for countries with more than 100 cities observed through the lens of Instagram and Twitter.

Country Visitors are more similar with each other in terms of POI preferences than with Foreign Visitors.

Figures 3(a-b) show the $LogRatio(Con)$ for countries with more than 100 cities observed in the two platforms. We can see that users from different countries have different similarities when visiting the same city; and that the similarity of users from the same country varies across visited cities. These observations highlight the need for recommendation methods that can account for the variability caused by user residence country as well as visiting cities. Interestingly, all countries with $LogRatio(Con) < 0$ in both Figures 3(a,b) are the ones whose capital cities are visited, indicating that in visiting the capital city of their own countries, Country Visitors are less similar than Foreign Visitors. We find that this is due to that Country Visitors are mostly commuters in visiting their capital cities, i.e. they go to work places in the capital cities.

Observation 2: *City Visitors are more similar with each other in terms of POI preferences than with Domestic Visitors.*

The distribution of $LogRatio(Cit)$ for all cities in all countries is shown in Figures 2(c-d) for Instagram and Twitter. We can observe that all cities have $LogRatio(Cit)$ greater than 0; 88.44% of them in Instagram and 88.15% in Twitter have $Sim(Cit^w)$ significantly larger than $Sim(Cit^a)$ (p-value < 0.01). We therefore conclude that the similarity within City Visitors is higher than that with Domestic Visitors. Comparing the distribution of cities with that of countries, all cities have $LogRatio(Con) > 0$ while there are some countries with $LogRatio(Con) < 0$ (those whose capital cities are visited), indicating that users from the same city are more similar than users from the same country. Moreover, we find that generally cities have larger values of $LogRatio$ than countries. For instance the mean values of the distribution of Amsterdam in Figures 2(a,c) are 4.09 and 5.07, respectively. This observation hints that cities generally have larger influence than countries on their residents' preferences.

4. RECURSIVE REGULARIZATION FOR MODELING FEATURE CO-INFLUENCE

We adopt the regularization technique to model the influence of auxiliary features. To do so, we have to consider feature relationships, and further allow for the learning of feature influence from historical user-item interaction data. For this we introduce a novel regularization method, named *recursive regularization*, that models the co-influence of features by recursively weighting each feature influence, traversing from root to leaves in the feature hierarchy.

4.1 Preliminaries

We first introduce the notations. Let $\mathcal{U} = \{u_1, u_2, \ldots, u_m\}$ be the set of m users, and $\mathcal{V} = \{v_1, v_2, \ldots, v_n\}$ be the set of n items. Given a user-item interaction matrix $\mathbf{R} \in \mathbb{R}^{m \times n}$, $\mathbf{R}_{ij}$ is a positive number denoting the rating given by u_i to v_j. $\mathbf{O} \in \mathbb{R}^{m \times n}$ denotes the indicator matrix, where $\mathbf{O}_{ij} = 1$ indicates that u_i rates v_j, and $\mathbf{O}_{ij} = 0$ otherwise. $\mathcal{F} = \{F_1, F_2, \ldots, F_t\}$ is the set of features, each of which describes at least one user in $\mathcal{U}$.

Table 2: Notations.

Notation	Explanation
$\mathcal{U}$,$\mathcal{V}$	user, item set
u_i/u_k, v_j	the ith/kth user in $\mathcal{U}$, and jth item in $\mathcal{V}$
$\mathbf{R}_{ij}$	rating given by user u_i to item v_j
$\hat{\mathbf{R}}_{ij}$	estimated rating for user u_i to item v_j
$\mathbf{O}$	indicator matrix indicating missing entries in $\mathbf{R}$
$\mathbf{U}_i$, $\mathbf{V}_j$	latent factors of user u_i and item v_j
$\mathcal{F}$	hierarchically-organized feature set
F	feature in the hierarchy
$Dis(F)$	regularization induced by *isolated* feature F
$Fu(F)$	feature unit with parent node F
$\mathbf{I}'(F)$	regularization by *isolated* feature unit $Fu(F)$
g, s	weighting parameters in propagating feature influence
$\mathbf{I}(F)$	regularization by feature unit $Fu(F)$ in hierarchy
$\mathbf{I}(\mathcal{F})$	regularization by feature hierarchy $\mathcal{F}$
$\mathbf{C}_{ik}$	regularization coefficient between $\mathbf{U}_i$ and $\mathbf{U}_k$
α	impact of recursive regularization
λ	regularizaton coefficient to avoid over-fitting
$\mathcal{J}$	objective function of ReMF framework

The features are organized hierarchically in a tree structure, where each node represents a feature in $\mathcal{F}$. The edge between a parent node $F_p \in \mathcal{F}$ and a child node $F_c \in \mathcal{F}$ represents a directed affiliation relationship, i.e. F_c belongs to F_p. Figure 4a shows an example containing three leaf features F_1, F_2, F_3, i.e. features with no children. F_1, F_2 are children of the internal feature F_4. F_3 and F_4 are children of the root feature F_5. For simplicity, we assume that each user is explicitly associated with at most one leaf feature in $\mathcal{F}$. Table 2 summarizes all the notations throughout this paper.

Our method is built on matrix factorization (MF) [15], which assumes the existence of latent structures in the user-item interaction matrix. By uncovering latent factors of users and items, it approximates the observed ratings and estimates the unobserved ratings. MF solves the following optimization problem:

$$\min_{\mathbf{U},\mathbf{V}} \frac{1}{2}\sum_{i,j}\mathbf{O}_{ij}(\mathbf{R}_{ij}-\mathbf{U}_i\mathbf{V}_j^T)^2+\frac{\lambda}{2}(\|\mathbf{U}\|_F^2+\|\mathbf{V}\|_F^2), \quad (1)$$

where $\mathbf{U} \in \mathbb{R}^{m\times d}$ and $\mathbf{V} \in \mathbb{R}^{n\times d}$ are the latent factors of users and items, respectively. d is the dimension of latent factors. λ is the regularization coefficient to avoid over-fitting. The unobserved rating for user u_i to item v_j can be estimated by the inner product of the corresponding user and item latent factors, i.e. $\hat{\mathbf{R}}_{ij} = \mathbf{U}_i^T\mathbf{V}_j$.

4.2 Modeling Influence of Feature Hierarchy on User-item Interactions

Step by step, we model the influence from a single feature to the combinations of features and finally the entire feature hierarchy.

Influence of an Isolated Feature. To start, we first define the regularization by an isolated feature F_p in the hierarchy as:

$$Dis(F_p) = \sum_{u_i,u_k\in F_p, i<k}\|\mathbf{U}_i-\mathbf{U}_k\|_F^2, \quad (2)$$

where $\|\mathbf{U}_i-\mathbf{U}_k\|_F^2$ is the squared Frobenius norm distance between the latent factors of u_i and u_k characterized by feature F_p: F_p poses regularization on the cumulation of the pairwise distance between users associated with it. Thus, $Dis(F_p)$ can be considered as the influence of the isolated feature F_p on user-item interactions by regularizing user latent factors. The definition here only considers the influence of an *isolated* feature, while the co-influence of the feature hierarchy contributed by the feature, i.e. influence of the feature *in the hierarchy*, is different from – but based on – the influence of the *isolated* feature, which will be illustrated later.

Our method models feature influence by regularizing user latent factors, and can be straightforward transferred to modeling the influence by regularizing item latent factors, or both of them.

Influence of an Isolated Feature Unit. Given the above definition, we now model the influence of an isolated combination of features, on learning user latent factors, by introducing the most important relationship among features in a hierarchy, i.e. *parent-child* relationship, based on which other relationships among features in the hierarchy such as *siblings*, *ancestors* can be derived. We first define the *feature unit*, i.e. $Fu(F_p)$, as the combination of a single parent node F_p and its children nodes, namely:

$$Fu(F_p) = \{F_p\} \cup \{F_c | \forall F_c \in children(F_p)\}.$$

Two examples of feature units $Fu(F_5)$ and $Fu(F_4)$ are shown in the red dash boxes in Figure 4a.

Then we consider the influence of an isolated feature unit on learning user latent factors by regularization. For each isolated feature unit $Fu(F_p)$, we denote its influence as $\mathbf{I}'(F_p)$, and assign it two parameters g_p, s_p, with the constraint $g_p + s_p = 1$. Parameters g_p and s_p are used to distribute the influence of the feature unit to two parts. One is given by the parent node, weighted by g_p, and the other is given by the children nodes, weighted by s_p. The influence of the isolated feature unit, i.e. $\mathbf{I}'(F_p)$, is then defined as:

$$\mathbf{I}'(F_p) = g_p Dis(F_p) + s_p(\sum_{\forall F_c\in children(F_p)} Dis(F_c)).$$

For example, the influence of the isolated feature unit $Fu(F_5)$ in Figure 4a, i.e. $\mathbf{I}'(F_5)$, is determined by both the influence of the parent node F_5, i.e. $Dis(F_5)$, weighted by g_5, and the influence of its children nodes, i.e. $Dis(F_3)$ and $Dis(F_4)$, weighted by s_5. The overall influence of this isolated feature unit is: $\mathbf{I}'(F_5) = g_5 Dis(F_5) + s_5(Dis(F_3) + Dis(F_4))$. Compared with the influence of the isolated feature F_5, the influence of feature F_5 in $Fu(F_5)$ is different, in that $Dis(F_5)$ is weighted by g_5.

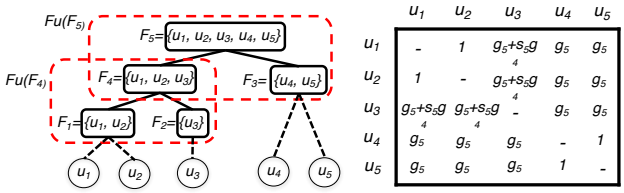
Influence of an Entire Feature Hierarchy. Based on the definition of the influence of an *isolated* feature unit, we now proceed to model the influence of feature unit *in the hierarchy*, thus to formally derive the overall influence of an entire feature hierarchy on user latent factors. Note that the influence of a feature unit *in the hierarchy* is different from – but based on – the influence of the *isolated* feature unit, and can be achieved by recursively defining the regularization of the feature unit *in the hierarchy*, given by:

DEFINITION 1 (**RECURSIVE REGULARIZATION**).

$$\mathbf{I}(F_p) = \begin{cases} g_p Dis(F_p) & +s_p(\sum_{\forall F_c\in children(F_p)} \mathbf{I}(F_c)), \\ & \textit{if } F_p \textit{ is an internal feature}; \\ Dis(F_p), & \textit{if } F_p \textit{ is a leaf feature and } |F_p| > 1; \\ 0, & \textit{otherwise}, \end{cases}$$

where $|F_p|$ is the number of users characterized by feature F_p.

From the above definition, we can see the difference between the influence of a feature unit *in the hierarchy* $\mathbf{I}(F_p)$ and the influence of an *isolated* feature unit $\mathbf{I}'(F_p)$, that is, $\mathbf{I}(F_p)$ is recursively defined on $\mathbf{I}(F_c)$. Put another way, the influence of a child feature is included in the influence of its parent feature. Hence, the influence of an entire feature hierarchy, denoted by $\mathbf{I}(\mathcal{F})$, is equivalent to that of the root feature, as it recursively includes the influence of all features in the hierarchy. As an example, Equation 3 shows the influence of the feature hierarchy in Figure 4a.



(a) Feature hierarchy

	u_1	u_2	u_3	u_4	u_5
u_1	-	1	$g_5+s_5g_4$	g_5	g_5
u_2	1	-	$g_5+s_5g_4$	g_5	g_5
u_3	$g_5+s_5g_4$	$g_5+s_5g_4$	-	g_5	g_5
u_4	g_5	g_5	g_5	-	1
u_5	g_5	g_5	g_5	1	-

(b) Regularization coefficients

Figure 4: (a) illustrates a feature hierarchy, where features with children (i.e. F_5, F_4) are called *internal features*. Particularly, F_5 is also named *root feature*, whereas features without children are called *leaf features*. Dash and solid lines respectively represent the user-feature (i.e. a user is characterized by a feature) and feature-feature (i.e. parent-child) relationships. Features in a red dash box comprises a feature unit. (b) shows the corresponding regularization coefficients of the corresponding example.

$$\begin{aligned}&\mathbf{I}(\mathcal{F}) = \mathbf{I}(F_5)\\ =&g_5 Dis(F_5) + s_5(\mathbf{I}(F_4) + \mathbf{I}(F_3))\\ =&g_5 Dis(F_5) + s_5(g_4 Dis(F_4) + s_4(\mathbf{I}(F_1) + \mathbf{I}(F_2)) + Dis(F_3))\\ =&g_5 Dis(F_5) + s_5(g_4 Dis(F_4) + s_4 Dis(F_1) + Dis(F_3)). \quad (3)\end{aligned}$$

The deduction of recursive regularization of a feature hierarchy is shown in Algorithm 1, where the co-influence of features is modeled as a regularization function parameterized by the weights of each feature in the hierarchy. These weights characterize the influence of distinct features, and can be further learnt from historical user-item interaction data, as we introduce in the next section.

Remark. By recursively weighting and combining feature influence over a hierarchy from the root feature to the leaves, recursive regularization can model the influence of an arbitrarily deep feature hierarchy that can be either balanced or imbalanced.

5. REMF: A RECOMMENDATION FRAMEWORK INTEGRATED WITH RECURSIVE REGULARIZATION

We first introduce a novel recommendation framework `ReMF`, that integrates the recursive regularization into the MF model to exploit auxiliary feature hierarchy. Then an optimization method and the complexity analysis for `ReMF` are presented.

5.1 The ReMF Framework

By incorporating recursive regularization into the MF, the `ReMF` framework is defined by:

DEFINITION 2 (**THE REMF FRAMEWORK**).

$$\min_{\substack{\mathbf{U},\mathbf{V},\\ g_p,s_p\forall F_p\in\mathcal{F}}} \mathcal{J} = \frac{1}{2}\sum_{i,j}\mathbf{O}_{ij}(\mathbf{R}_{ij}-\mathbf{U}_i\mathbf{V}_j^T)^2+\frac{\alpha}{2}\mathbf{I}(\mathcal{F})+\frac{\lambda}{2}(\|\mathbf{U}\|_F^2+\|\mathbf{V}\|_F^2)$$

where α is a regularization parameter that controls the impact of recursive regularization, i.e. $\mathbf{I}(\mathcal{F})$.

Thanks to recursive regularization, `ReMF` can model the co-influence of features in the hierarchy to learn user latent factors.

It also characterizes the distinct influence of each feature, thus helping with the interpretation of the effect of each feature in the hierarchy on recommendation, illustrated as follows.

Considering the example of Figure 4, based on Equations 2 and 3, the feature hierarchy influence $\mathbf{I}(\mathcal{F})$ can be rewritten as:

$$(g_5 + s_5 g_4 + s_5 s_4)\|\mathbf{U}_1 - \mathbf{U}_2\|_F^2 + (g_5 + s_5 g_4)\|\mathbf{U}_1 - \mathbf{U}_3\|_F^2 + \ldots,$$

where the strength of the regularization between u_1, u_2's latent factors is $(g_5 + s_5 g_4 + s_5 s_4)$, and that of u_1, u_3's latent factors is $(g_5 + s_5 g_4)$. In fact, the strength of regularization is the combination of influence of different features. For simplicity, we assume $g = s = 0.5$ for each internal feature. Therefore, the strength of regularization between u_1, u_2's latent factors is $(g_5 + s_5 g_4 + s_5 s_4) = 1$, from which we could see that the feature F_5 has an influence of $g_5 = 0.5$, while its children features F_4 and F_1 have influence of $s_5 g_4 = 0.25$ and $s_5 s4 = 0.25$, respectively. Then, for u_1, u_3, the strength of regularization between their latent factors is $(g_5 + s_5 g_4) = 0.75$, where the features F_5, F_4 have influence of $g_5 = 0.5, s_5 g_4 = 0.25$, respectively. The distinct influence of features on learning user latent factors can therefore be characterized by certain functions of the weights (g, s).

Algorithm 1: Recursive Regularization Deduction

Input: feature hierarchy $\mathcal{F}$, $g_p, s_p \forall F_p \in \mathcal{F}$
foreach $F_p \in \mathcal{F}$ **do**
  $\mathbf{I}(F_p) \leftarrow 0$;
$layer \leftarrow$ #layers of $\mathcal{F}$;
for $l = 0; l \leq layer; l++$ **do**
  foreach *feature* F_p *at layer* l *of* $\mathcal{F}$ **do**
    if F_p *is a leaf feature* ($l = 0$) *and* $|F_p| > 1$ **then**
      $\mathbf{I}(F_p) \leftarrow Dis(F_p)$;
    else if F_p *is an internal feature* ($l \neq 0$) **then**
      $\mathbf{I}(F_p) \leftarrow g_p Dis(F_p) + s_p(\sum_{\forall F_c \in children(F_p)} \mathbf{I}(F_c))$;
$\mathbf{I}(\mathcal{F}) \leftarrow \mathbf{I}(F_{root})$;

To formally derive feature influence on an arbitrary pair of users, we define the *regularization coefficient* $\mathbf{C}_{ik}$ to represent the strength of regularization between u_i and u_k, where a greater value of $\mathbf{C}_{ik}$ indicates a higher correlation between the two users. Hence, $\mathbf{I}(\mathcal{F})$ can be reformulated as:

$$\mathbf{I}(\mathcal{F}) = \sum_{u_i,u_k\in\mathcal{U},i<k} \mathbf{C}_{ik}\|\mathbf{U}_i - \mathbf{U}_k\|_F^2,$$

We next introduce two theorems for deriving $\mathbf{C}_{ik}$, which is the combination of the influence by different features on u_i and u_k.

THEOREM 1. *The regularization coefficient for any pair of users u_i, u_k (i.e. $\mathbf{C}_{ik}$) characterized by the same leaf feature is 1:*

$$g_{root} + s_{root}(g_{c_1} + s_{c_1}(g_{c_2} + s_{c_2}(\ldots(g_{c_l} + s_{c_l})))) = 1,$$

where the list $\{F_{root}, F_{c_1}, F_{c_2}, \ldots, F_{c_l}\}$ is the set of the common features of u_i and u_k, ordered in a sequence from the root feature F_{root} to the leaf feature F_{c_l}.

Proof. This is straightforward to prove, due to the constraint $g+s = 1$. Considering the example $\{u_1, u_2\}$ in Figure 4, the sum of the relevant regularization terms, i.e. $g_5 Dis(F_5), s_5 g_4 Dis(F_4)$ and $s_5 s_4 Dis(F_1)$, in Equation 3 is:

$$\begin{aligned}&(g_5 + s_5(g_4 + s_4))\|\mathbf{U}_1 - \mathbf{U}_2\|_F^2\\ =&(g_5 + s_5)\|\mathbf{U}_1 - \mathbf{U}_2\|_F^2 = \|\mathbf{U}_1 - \mathbf{U}_2\|_F^2.\end{aligned}$$

THEOREM 2. *For any pair of users u_i, u_k not characterized by a common leaf feature, the regularization coefficient (i.e. $\mathbf{C}_{ik}$) is:*

$$g_{root} + s_{root}(g_{c_1} + s_{c_1}(g_{c_2} + s_{c_2}(\ldots(g_{c_l})))),$$

where the list $\{F_{root}, F_{c_1}, F_{c_2}, \ldots, F_{c_l}\}$ is the set of the common features of u_i and u_k, ordered from the root feature F_{root} to the deepest common feature F_{c_l}.

Proof. All possible features that can influence the regularization coefficient of u_i, u_k are their deepest common feature, and the parents and ancestors of the deepest common feature.

According to the above theorems, the value of regularization coefficient always falls into the range of $[0, 1]$, with 1 indicating the

Algorithm 2: ReMF Model Learning

Input: rating matrix $\mathbf{R}$, feature hierarchy $\mathcal{F}$, $d, \gamma, \lambda, \alpha, iter$

Initialize $\mathbf{U}, \mathbf{V}, g_p, s_p$, and $\forall F_p \in \mathcal{F}$;
for $t = 1; t \leq iter; t++$ **do**
  foreach $\mathbf{U}_i \in \mathbf{U}$, $\mathbf{V}_j \in \mathbf{V}$ **do**
    $\mathbf{U}_i^{(t)} \leftarrow \mathbf{U}_i^{(t-1)} - \gamma \frac{\partial \mathcal{J}}{\partial \mathbf{U}_i}$;
    $\mathbf{V}_j^{(t)} \leftarrow \mathbf{V}_j^{(t-1)} - \gamma \frac{\partial \mathcal{J}}{\partial \mathbf{V}_j}$;
  foreach *Internal feature in the hierarchy* **do**
    $g_p^{(t)} \leftarrow g_p^{(t-1)} - \gamma \frac{\partial \mathcal{J}}{\partial g_p}$;
    $s_p^{(t)} \leftarrow s_p^{(t-1)} - \gamma \frac{\partial \mathcal{J}}{\partial s_p}$;
  Calculate $\mathcal{J}$ by Algorithm 1 and Definition 2;
  if $\mathcal{J}$ *has converged* **then**
    break;

full regularization and 0 indicating no regularization. As an example, Figure 4b shows the regularization coefficients of the feature hierarchy in Figure 4a.

These regularization coefficients naturally connect ReMF to network based recommendation methods, which also consider pairwise regularization on users. There are however two essential differences: 1) network-based regularization coefficients are usually hard-coded, while our regularization coefficients are modeled from the feature hierarchy structure, and expressed by the function of weights (g, s). And, 2) (g, s), which parametrizes the distinct feature influence while being automatically learnt from the historical user-item interaction data, as we will address in the next subsection.

5.2 The Optimization Method for ReMF

We adopt the stochastic gradient descent scheme [14, 15] to optimize our objective function.

Updating $\mathbf{U}, \mathbf{V}$. The gradients of $\mathbf{U}_i, \mathbf{V}_j$ are given by:

$$\frac{\partial \mathcal{J}}{\partial \mathbf{U}_i} = -\sum_j \mathbf{O}_{ij}(\mathbf{R}_{ij} - \mathbf{U}_i \mathbf{V}_j^T)\mathbf{V}_j + \lambda \mathbf{U}_i + \alpha \sum_{u_i, u_k \in \mathcal{U}, i<k} \mathbf{C}_{ik}(\mathbf{U}_i - \mathbf{U}_k),$$

$$\frac{\partial \mathcal{J}}{\partial \mathbf{V}_j} = -\sum_i \mathbf{O}_{ij}(\mathbf{R}_{ij} - \mathbf{U}_i \mathbf{V}_j^T)\mathbf{U}_i + \lambda \mathbf{V}_j.$$

Updating (g, s). (g, s) can be predefined heuristically, or handcrafted by domain experts who can fairly quantify the influence of different features. Instead, we provide an effective data-driven solution that automatically learns (g, s) based on the historical user-item interaction data.

We only need to estimate (g, s) for internal features in the hierarchy, since the leaf features do not have children. For an internal feature F_p, the gradients of g_p, s_p are equivalent to the multipliers of g_p, s_p in $\mathbf{I}(\mathcal{F})$. Thus, we have:

$$\frac{\partial \mathcal{J}}{\partial g_p} = \begin{cases} Dis(F_p), & \text{if } F_p \text{ is root,} \\ \prod_{\forall a: F_a \in ancestors(F_p)} s_a Dis(F_p), & \text{otherwise;} \end{cases}$$

$$\frac{\partial \mathcal{J}}{\partial s_p} = \begin{cases} \sum_{\forall F_c \in children(F_p)} \mathbf{I}(F_c), & \text{if } F_p \text{ is root,} \\ \prod_{\forall a: F_a \in ancestors(F_p)} s_a (\sum_{\forall F_c \in children(F_p)} \mathbf{I}(F_c)), & \text{otherwise.} \end{cases}$$

According to the constraint $g_p + s_p = 1$, we can update g_p (or s_p) using the gradient and the other by $s_p = 1 - g_p$ (or $g_p = 1 - s_p$). The detailed learning process is shown in Algorithm 2.

Complexity Analysis The computational time is mainly taken by evaluating the objective function $\mathcal{J}$ and updating the related variables. The time to compute the $\mathcal{J}$ is $\mathcal{O}(d|\mathbf{R}| + dm^2)$, where $|\mathbf{R}|$ is the number of non-zero observations in the rating matrix $\mathbf{R}$. For all gradients $\frac{\partial \mathcal{J}}{\partial \mathbf{U}_i}$, $\frac{\partial \mathcal{J}}{\partial \mathbf{V}_j}$, $\frac{\partial \mathcal{J}}{\partial g_p}$, $\frac{\partial \mathcal{J}}{\partial s_p}$, the computational time are $\mathcal{O}(d|\mathbf{R}| + dm^2)$, $\mathcal{O}(d|\mathbf{R}|)$, $\mathcal{O}\left(d \sum_{l=0}^{layer-1} \frac{\overline{m}_l(\overline{m}_l - 1)n_l}{2}\right)$ and $\mathcal{O}(|s_p|)$, respectively. Wherein $\overline{m}_l$ denotes the average number of users in each node at layer l, n_l denotes the number of nodes at layer l, and $|s_p|$ ($\ll |\mathbf{R}|$) denotes the number of internal nodes. Particularly, we leverage $s_p = (1 - g_p)$ to update s_p. The overall computational complexity of Algorithm 2 is $(\#iteration * \mathcal{O}(d|\mathbf{R}| + dq))$, where $q = max(\sum_{l=0}^{layer-1} \frac{\overline{m}_l(\overline{m}_l - 1)n_l}{2}, m^2)$. In real-world applications $\overline{m}_l$ is typically small (e.g. power-law distributed), thus making ReMF scalable to large data set.

6. EXPERIMENTS AND RESULTS

We assess the performance of ReMF with a comparison with the state-of-the-art, feature-based, hierarchy-based recommendation methods. The comparison is performed over 1) the data sets introduced in Section 3, for POI recommendation with user feature hierarchy; and 2) a data set from the Amazon Web store [19], for product recommendation with item feature hierarchy.

6.1 Experimental Setup

Evaluation. We adopt the standard 5-fold cross-validation, and the following 3 metrics for evaluation: Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) [13, 22] to measure the error of predicted ratings; and Area Under the ROC Curve (AUC) [9, 31] to measure the quality of predicted ranking of items (ranked according to the predicted ratings). The smaller MAE and RMSE, and the larger AUC, the better the recommendation performance.

Comparison Methods. The following methods are compared: (1) **MF** [15]: matrix factorization method; (2) **CMF** [25]: collective MF; (3) **TaxMF** [13]: taxonomy-based MF; (4) **SoReg** [18]: network-based recommendation method incorporating social relations; (5) **FM** [22]: factorization machine; (6) **HieFM**: factorization machine with hierarchy information.

HieFM is a variation of FM that considers each features path in the hierarchy (from root to leaf nodes) as an additional feature in the design vectors of FM. Similar to FM, CMF and TaxMF can also incorporate path-based features. As FM outperforms CMF and TaxMF (see Section 6.3), we limit our comparison with previous methods exploiting path-based features to HieFM.

Parameter Settings. We empirically set optimal parameters for each method using a grid search in $\{0.0001, 0.001, 0.01, 0.05\}$ for both λ (including 1-way and 2-way regularization of FM) and the learning rate γ; $\alpha = 0.5$ for CMF; $\beta = 0.01$ for SoReg. For fair comparison, we set $d = 10$ (the dimension of latent factors) for all the methods, and adopt all features (i.e. continent, country, and city) as input in TaxMF, CMF, FM and HieFM. HieFM has path-based features as additional hierarchy information. In SoReg, we model the social relations among users by counting the number of common features, under the assumption that the commonality establishes implicit social relationships based on the geo-social correlation phenomenon [8]. Without loss of generality, we adopt $f(x) = 1/(1 + x^{-1})$ to map each #check-in $\mathbf{R}_{ij} \in \mathbf{R}$ in POI data sets into the interval $(0, 1)$ [7].

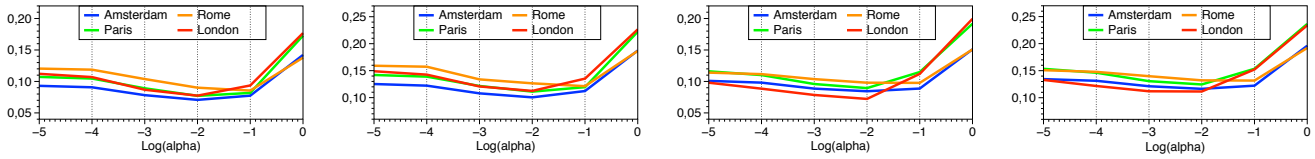
6.2 Results of ReMF

We analyze the influence of recursive regularization on ReMF performance, and discuss how the weighting parameters g, s can help the interpretation of recommendation results.

The Impact of α. In ReMF, α controls the strength of recursive regularization of feature hierarchy. We apply a grid search in $\{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 10^{0}\}$ to investigate the impact of α on recommendation performance. Results are shown in Figure 5. As α varies from small to large, the performance first increases then decreases, with the maximum reached at the range $[10^{-2}, 10^{-1}]$.

Table 3: Performance of the considered recommendation methods on the testing views "All" and "Cold start" of POI data sets. The best performance for each city is boldfaced; the runner up is labelled with "*". The improvements by the best method on all data sets are statistically significant (p-value < 0.01).

View		Data Set	MAE							RMSE						
			MF	**CMF**	**TaxMF**	**SoReg**	**FM**	**HieFM**	**ReMF**	**MF**	**CMF**	**TaxMF**	**SoReg**	**FM**	**HieFM**	**ReMF**
All	Inst.	Amsterdam	0.1957	0.1564	0.1426	0.1038	0.0876	0.0822*	**0.0707**	0.3134	0.1940	0.1934	0.1455	0.1373	0.1352*	**0.1005**
		Paris	0.1539	0.1550	0.1416	0.1208	0.0790*	0.0830	**0.0772**	0.2675	0.1921	0.1849	0.1825	0.1293	0.1184*	**0.1111**
		Rome	0.2549	0.1584	0.1474	0.1355	0.0912	0.0885*	**0.0855**	0.3860	0.1967	0.1859	0.1912	0.1403	0.1389*	**0.1212**
		London	0.1799	0.1559	0.1369	0.1250	0.0834*	0.0851	**0.0774**	0.2964	0.1934	0.1762	0.1840	0.1347*	0.1396	**0.1124**
	Twit.	Amsterdam	0.2264	0.1606	0.1345	0.1229	0.0989	0.0942*	**0.0844**	0.3473	0.1996	0.1717	0.1669	0.1540	0.1454*	**0.1164**
		Paris	0.2014	0.1714	0.1552	0.1266	0.0956	0.0935*	**0.0894**	0.3207	0.2136	0.2038	0.1687	0.1408	0.1387*	**0.1245**
		Rome	0.2681	0.1713	0.1591	0.1345	0.1023	0.0996*	**0.0977**	0.3902	0.2132	0.2030	0.1831	0.1534	0.1469*	**0.1317**
		London	0.2176	0.1659	0.1545	0.1122	0.0931	0.0898*	**0.0772**	0.3075	0.2065	0.1959	0.1540	0.1407	0.1375*	**0.1115**
Cold start	Inst.	Amsterdam	0.2938	0.1552	0.1457	0.1051	0.0924	0.0885*	**0.0712**	0.3877	0.1926	0.1904	0.1479	0.1443	0.1391*	**0.1040**
		Paris	0.1939	0.1541	0.1476	0.1173	0.0849	0.0848*	**0.0799**	0.3110	0.1907	0.1896	0.1713	0.1374	0.1287*	**0.1183**
		Rome	0.3840	0.1614	0.1518	0.1356	0.0952	0.0938*	**0.0808**	0.4868	0.1990	0.1925	0.1845	0.1455*	0.1506	**0.1250**
		London	0.3032	0.1544	0.1415	0.1221	0.0893*	0.0901	**0.0791**	0.3978	0.1917	0.1819	0.1685	0.1426	0.1425*	**0.1161**
	Twit.	Amsterdam	0.3261	0.1604	0.1426	0.1189	0.1006	0.0966*	**0.0849**	0.4003	0.1982	0.1832	0.1609	0.1558	0.1514*	**0.1172**
		Paris	0.2439	0.1706	0.1640	0.1271	0.1012	0.0945*	**0.0873**	0.3764	0.2123	0.2120	0.1713	0.1485*	0.1502	**0.1226**
		Rome	0.3922	0.1718	0.1681	0.1343	0.1073	0.1070*	**0.0988**	0.4951	0.2133	0.2136	0.1833	0.1559	0.1517*	**0.1359**
		London	0.3301	0.1642	0.1587	0.1128	0.0967	0.0924*	**0.0756**	0.3976	0.2043	0.2013	0.1563	0.1475	0.1436*	**0.1093**



(a) MAE - Instagram (b) RMSE - Instagram (c) MAE - Twitter (d) RMSE - Twitter

Figure 5: The effects of α on the performance of ReMF on Instagram and Twitter measured by MAE and RMSE.

The performance variations across data sets suggest the need for data set-specific settings; the similarity in performance variation across α values shows the robustness of ReMF.

Interpretation from (g, s). We examine (g, s) for the internal features, i.e. continents and countries, learnt from data. Table 4 shows the list of continents and countries ranked according to their g values. Recall that for a continent (country), $g > s$ means that the continent (country) has a stronger effect on user preferences than and its children features, i.e. countries (cities).

In general the continents have relatively smaller effects on user preferences (with g values all below 0.2), suggesting that continents have weaker effects than their countries. In addition, we observe a big variance in the g values of countries, indicating that different countries have different influence on user preferences. The high variance of countries' g values proves the necessity of parameterizing g, s in recommendation. We then compare the influence of countries and cities on their residents' preferences. As cities of a country and the country comprise a feature unit, the influence of a city can be measured by $s = 1 - g$, where g is the influence of the country. We can see from Table 4 that most countries have $g < 0.5$ (only 3 countries have $g > 0.5$), i.e. $s > 0.5$, indicating that the influence of cities in most countries have more influence on their residents' preferences than the countries themselves.

Table 4: Values of g for continents and top/bottom countries in the dataset.

Continents		**Top countries**		**Bottom countries**	
Name	g	Name	g	Name	g
Europe	0.1837	Portugal	0.6915	Chile	0.0211
America	0.1656	Monaco	0.5813	Thailand	0.0175
Asia	0.1534	Serbia	0.5130	Spain	0.0100
Africa	0.0375	Poland	0.4453	Indonesia	0.0081
Oceania	0.0139	Hungary	0.4141	Belgium	0.0064

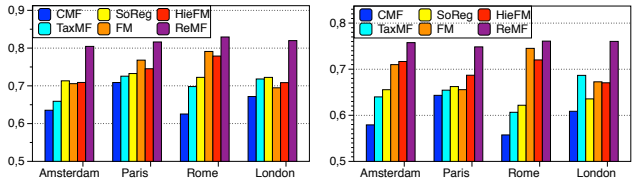
6.3 Comparative Results

Rating Performance. Two views are created for each data set: 1) the "All" view includes all users; while 2) the "Cold start" view indicates that only users with ≤ 5 ratings are involved in the test set. Table 3 compares the performance of the considered recommendation methods for all data sets. Unsurprisingly, the basic matrix factorization model is consistently outperformed by feature-based recommendation methods; this shows that, in the context of the targeted evaluation scenario, the usage of auxiliary information about users positively affects recommendation accuracy. In addition, FM outperforms CMF, TaxMF and SoReg. This could be explained by FM considering item-feature interactions, in addition to user-item and user-feature interactions.

HieMF in general outperforms FM, suggesting that information about feature relationships (paths) can help predicting user preferences. ReMF consistently outperforms the methods in the comparison pool, with an average performance gain (w.r.t. the second best method) of **7.20%** (MAE) and **15.07%** (RMSE). Paired t-test shows that the improvements of ReMF on all data sets are significant (p-value < 0.01). Such big improvements clearly show the effectiveness of recursive regularization, and the advantage derived from the full inclusion of information about feature relationships.

Table 3 (data view "Cold start") reports the results with cold start users. As in the previous case, ReMF achieves the best performance compared with other methods, and significantly outperforms the second best methods in all data sets (p-value < 0.01) by **12.02%** and **17.53%** w.r.t. MAE and RMSE respectively. The relatively larger improvements on the testing view "Cold start" than on "All" indicates that ReMF has higher capability in coping with the cold start problem compared to the state-of-the-art methods.

Ranking Performance. We further evaluate the ranking quality of items recommended by ReMF and other methods in the comparison pool. Results are shown in Figures 6(a-b) for data sets from Instagram and Twitter, respectively. ReMF significantly outperforms the second best method (p-value < 0.01) on all data sets by **9.86%** on average, reaching an averaged AUC of 0.8175 in Instagram and 0.7568 in Twitter. These observations show that the influence of feature hierarchy modeled by recursive regularization can effectively complement user-item interaction data in ranking prediction.



(a) AUC - Instagram (b) AUC - Twitter

Figure 6: AUC of ReMF and the comparative methods on POI data sets of four cities, through the lens of (a) Instagram and (b) Twitter.

Generalizability. We test the performance of ReMF on another task, i.e. product recommendation, using the data from Amazon web store [19]. Different from the POI data sets, here we consider the feature hierarchy of items. We focus on the product category of "Clothing, Shoes & Jewelry", having maximal depth of 7, and an unbalanced feature hierarchy. An example path in the hierarchy from the root feature to the leaf is "Clothing, Shoes & Jewelry $\rightarrow$ Men $\rightarrow$ Accessories $\rightarrow$ Wallets". We uniformly sample the raw data set to include $100,810$ ratings performed by $34,817$ users to $45,716$ items. Table 5 compares the performance of ReMF and the other methods in the comparison pool, measured by RMSE, which is more indicative of large errors than MAE. As in the previous setting, ReMF significantly outperforms the second best method (p-value < 0.01), i.e. HieFM, by **5.46%** on the testing view of "All" and **7.42%** on "Cold start". These results show that ReMF can be effective in multiple recommendation tasks, and with different topologies of features hierarchy.

Table 5: Performance (RMSE) on the testing views "All" and "Cold start" of Amazon data set. The best performance is boldfaced; the runner up is labelled with "*". All improvements by the best method are statistically significant (p-value < 0.01).

	CMF	TaxMF	SoReg	FM	HieFM	ReMF
All	1.6356	1.3921	1.3912	1.3899	1.3847*	**1.3091**
Cold start	1.6386	1.4057	1.4054	1.4074	1.4033*	**1.3242**

7. CONCLUSIONS

Hierarchies are a common way to capture relationships between features. Yet, the value of this additional information is not fully exploited by state-of-the-art feature-based recommendation methods. This paper proposes a novel regularization method named recursive regularization for modeling the co-influence of features in the hierarchy on user-item interactions. Based on this, a new recommendation framework ReMF is proposed to learn hierarchical feature influence from historical user-item interaction data. Experimental validation on real-word data sets shows that ReMF can largely outperform state-of-the-art methods, proving the value residing in the exploitation of feature hierarchies for better learning user and item latent factors.

We stress how recursive regularization does not only apply to tree-like data structures (hierarchy), but also to a forest of trees: adding a root feature transforms a set of trees to one tree. Generalization to graphs is less trivial, and therefore left to future work.

8. REFERENCES

[1] D. Agarwal and B.-C. Chen. Regression-based latent factor models. In *KDD*, 2009.

[2] A. Albadvi and M. Shahbazi. A hybrid recommendation technique based on product category attributes. *Expert Systems with Applications*, 36(9):11480–11488, 2009.

[3] S. Bocconi, A. Bozzon, A. Psyllidis, C. Titos Bolivar, and G.-J. Houben. Social glass: A platform for urban analytics and decision-making through heterogeneous social data. In *WWW Companion*, 2015.

[4] T. Chen, W. Zhang, Q. Lu, K. Chen, Z. Zheng, and Y. Yu. Svdfeature: a toolkit for feature-based collaborative filtering. *Journal of Machine Learning Research*, 13(1):3619–3622, 2012.

[5] Z. Cheng, J. Caverlee, K. Lee, and D. Z. Sui. Exploring millions of footprints in location sharing services. In *ICWSM*, 2011.

[6] Z. Gantner, L. Drumond, C. Freudenthaler, S. Rendle, and L. Schmidt-Thieme. Learning attribute-to-feature mappings for cold-start recommendations. In *ICDM*, 2010.

[7] H. Gao, J. Tang, X. Hu, and H. Liu. Content-aware point of interest recommendation on location-based social networks. In *AAAI*, 2015.

[8] H. Gao, J. Tang, and H. Liu. gscorr: modeling geo-social correlations for new check-ins on location-based social networks. In *CIKM*, 2012.

[9] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems*, 22(1):5–53, 2004.

[10] Z. Hu, P. Huang, Y. Deng, Y. Gao, and E. P. Xing. Entity hierarchy embedding. In *ACL-IJCNLP*, 2015.

[11] R. Jenatton, J. Mairal, F. R. Bach, and G. R. Obozinski. Proximal methods for sparse hierarchical dictionary learning. In *ICML*, 2010.

[12] S. Kim and E. P. Xing. Tree-guided group lasso for multi-task regression with structured sparsity. In *ICML*, 2010.

[13] N. Koenigstein, G. Dror, and Y. Koren. Yahoo! music recommendations: modeling music ratings with temporal dynamics and item taxonomy. In *ResSys*, 2011.

[14] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *KDD*, 2008.

[15] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, (8):30–37, 2009.

[16] W.-J. Li and D.-Y. Yeung. Relation regularized matrix factorization. In *IJCAI*, 2009.

[17] C. Lippert, S. H. Weber, Y. Huang, V. Tresp, M. Schubert, and H.-P. Kriegel. Relation prediction in multi-relational domains using matrix factorization. In *NIPS Workshop SISO*, 2008.

[18] H. Ma, D. Zhou, C. Liu, M. R. Lyu, and I. King. Recommender systems with social regularization. In *WSDM*, 2011.

[19] J. McAuley, C. Targett, Q. Shi, and A. van den Hengel. Image-based recommendations on styles and substitutes. In *SIGIR*, 2015.

[20] Y. Moshfeghi, B. Piwowarski, and J. M. Jose. Handling data sparsity in collaborative filtering using emotion and semantic based features. In *SIGIR*, 2011.

[21] S. Rendle. Factorization machines. In *ICDM*, 2010.

[22] S. Rendle, Z. Gantner, C. Freudenthaler, and L. Schmidt-Thieme. Fast context-aware recommendations with factorization machines. In *SIGIR*, 2011.

[23] F. Ricci, L. Rokach, and B. Shapira. *Introduction to recommender systems handbook*. Springer, 2011.

[24] Y. Shi, M. Larson, and A. Hanjalic. Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges. *ACM Computing Surveys*, 47(1):3, 2014.

[25] A. P. Singh and G. J. Gordon. Relational learning via collective matrix factorization. In *KDD*, 2008.

[26] J. Tang, H. Gao, X. Hu, and H. Liu. Exploiting homophily effect for trust prediction. In *WSDM*, 2013.

[27] J. Tang, X. Hu, and H. Liu. Social recommendation: a review. *Social Network Analysis and Mining*, 3(4):1113–1133, 2013.

[28] S. Wang, J. Tang, Y. Wang, and H. Liu. Exploring implicit hierarchical structures for recommender systems. In *IJCAI*, 2015.

[29] L.-T. Weng, Y. Xu, Y. Li, and R. Nayak. Exploiting item taxonomy for solving cold-start problem in recommendation making. In *ICTAI*, 2008.

[30] M. Ye, P. Yin, W.-C. Lee, and D.-L. Lee. Exploiting geographical influence for collaborative point-of-interest recommendation. In *SIGIR*, 2011.

[31] T. Zhao, J. McAuley, and I. King. Improving latent factor models via personalized feature projection for one class recommendation. In *CIKM*, 2015.

[32] Y. Zhen, W.-J. Li, and D.-Y. Yeung. Tagicofi: tag informed collaborative filtering. In *RecSys*, 2009.

[33] C.-N. Ziegler, G. Lausen, and L. Schmidt-Thieme. Taxonomy-driven computation of product recommendations. In *CIKM*, 2004.