Structured Sparse Regression for Recommender Systems

Mingjie Qian[†], Liangjie Hong[§], Yue Shi[§], Suju Rajan[§]
[†] Department of Computer Science, University of Illinois at Urbana-Champaign, IL, USA
[§] Personalization Sciences, Yahoo Labs, CA, USA
mqian2@illinois.edu, {liangjie,yueshi,suju}@yahoo-inc.com

ABSTRACT

Feature-based collaborative filtering models, such as state-of-the-art factorization machines and regression-based latent factor models, rarely consider features' structural information, ignoring the heterogeneity of inter-type and intra-type relationships. Naïvely treating all feature pairs equally would potentially deteriorate the overall recommendation performance. In addition, human prior knowledge and other hierarchical or graphical structures are often available for some features, e.g., the country-state-city hierarchy for geographic features and the topical taxonomy for article features. It is a challenge to utilize the prior knowledge to further boost performance of state-of-the-art models. In this paper we employ rich features from both user and item sides to enhance latent factors learnt from interaction data, uncovering hidden structures from features' relationships and learning sparse pairwise and tree structural connections among features. Our framework borrows the modeling strengh from both structural sparsity modeling and latent factor models. Experiments on a real-world large-scale recommendation data set demonstrated that the proposed model outperforms several strong state-of-the-art baselines.

Categories and Subject Descriptors

H.3.5 [**Information Storage and Retrieval**]: Online Information Services

Keywords

Structured Sparse Regression, Structured Sparse Feature Graph Learning, Hierarchical Sparse Coding, Structured Sparse Coding, Feature-based Collaborative Filtering

1. INTRODUCTION

Feature based latent factor models have received increasing attention in recent years due to its capability to effectively solve the cold-start problem. There have been many feature based collaborative filtering (CF) models proposed recently, which can be grouped into two categories. The first type of models includes all variants of latent factor models (LFM) which have been proven as an effective approach to personalization and recommender systems. The core of LFM is to learn user-specific and item-specific features from user-item interactions and utilize these features for future predictions/recommendations. State-of-the-art LFM exploits low-rank latent spaces of users and features and treats latent factors that are learnt from user-item historical data as features. This type of models has gained significant successes in a number of applications, including the Netflix competition. The second category is the factorization machine (FM), which explicitly learns the mapping function from features to rating score circumventing the dependency on user/item latent factors as in the latent factor models, resulting in an effective model for the cold start problem [9].

Although these feature-based CF models have been shown to be effective, they do not utilize the feature structure information. For example, conventional latent factor models (e.g., matrix factorization or tensor factorization models) like RLFM [1, 2] learn mapping functions from user/item features to user/item latent vectors assuming the features have a flat first-order structure. Later, [10] showed that this kind of mapping can be extended to any non-linear models. Although the formalism is flexible, it leaves too much room for practitioners to choose which non-linear model to use for a particular application. Also, it is hard to incorporate human prior knowledge on the feature structure into the framework, unless through careful feature engineering, and the proposed inference algorithm is difficult to use in large-scale settings. Similar to RLFM, Gantner et al. [4] proposed a model to explicitly learn the mapping function from features to latent factors, resulting in an effective model for the cold start problem. But it still makes the flat first-order feature structure assumption. In the other line of work, Rendle et al. [9] proposed a more compact model called factorization machine FM. Basically FM is a second-order regression model which directly maps the user-item-event concatenated features to rating score by learning the implicit mapping functions from features to latent factors, resulting in an effective model for the cold start problem. However, the issue of encoding structural human prior information still boils down to sophisticated feature engineering and it's not clear how to incorporate the heterogenous feature structures into model training to enhance the rating prediction performance. Though FM considers the second-order feature structure, it simply uses all the feature pairs for prediction. Quite

CIKM'15, October 19–23, 2015, Melbourne, VIC, Australia
Copyright 2015 ACM 978-1-4503-3794-6/15/10 ...$15.00.
DOI: http://dx.doi.org/10.1145/2806416.2806641.

a lot of work in sparse coding area have shown that many signals tend to have a sparse representation from basic components in nature, and a sparse model often outperforms a dense model and also has the variable selection effect. Inspired by this, a sparse model that uses an appropriate subset of feature pairs might have a better performance.

In practices, human prior knowledge or explicit structure information about these features is also sometimes available. For example, the topical categories on news articles may naturally be organized into hierarchies or graphs. Another good example would be demographical information about users, especially their geo-locations that are aligned with countries, states and cities, defined in real-world geo-political settings. These prior knowledge and structures are invaluable information for better user understanding and profiling and eventually better recommendation results. However, it is not straightforward to encode this kind of structural prior knowledge into state-of-the-art recommendation models like `RLFM` and `FM`. One approach might be to construct features capturing these structures and embed them into regression models. But the interplay between an optimal way to construct such features and train a better regression model based on these features to map to latent features becomes non-trivial in this case. Some previous work has been proposed to impose structural information on latent variable models, which are not necessarily directed graphical models. For instance, He et al. [5] proposed a general learning framework which induces sparsity on the undirected graphical model imposed on the vector of latent factors. Although the paper shares a similar idea with our framework, their work cannot handle heterogeneous types of features and complex dependencies. Also, it is hard to link their work to state-of-the-art `LFM` used in `CF` settings. Along the line of undirected graphical models, Min et al. [8] proposed sparse high-order Boltzmann machines, aiming to capture dependencies between latent variables. Again, it is not obvious to plug the model into state-of-the-art `CF` approaches.

In this paper, we propose a structured sparse second-order regression model with structural prior knowledge in a principled way. The notion of types of features is introduced such that different types of features would have different structures (e.g., topical categories versus geographical locations). We consider two kinds of structures. For inter-typed features (which are of different kinds), the model is able to learn sparse relationships between different types of features (e.g., age and gender). For intra-typed features (which are in the same kind) that have a hierarchy (tree), e.g., we have the country-state-city hierarchical tree for the geo-location feature terms, the model learns a sparse hierarchical structure on the tree such that if a parent feature edge (or interchangeably a feature pair) is selected, then all its descendant edges should be selected as well, and if a parent edge is removed then all its descendant edges should be removed too.

2. STRUCTURED SPARSE REGRESSION

We model the rating score of an event by a user on an item by a second-order polynomial regression

$$\hat{y} = b + \langle \mathbf{w}, \mathbf{x} \rangle + \sum_{i=1}^{p} \sum_{j=i+1}^{p} \alpha_{ij} x[i] x[j], \tag{1}$$

where b is the global bias for all events, $\mathbf{w}$ is the first-order weights, and p is the feature size. $\mathbf{x}$ is the concatenated feature vector for the user-item-event triplet as is used in `FM`. Apparently, we directly use a pair-wise weight factor α_{ij} for each feature pair (or edge) (i, j) rather than the inner product of latent factors $\mathbf{v}_i$ and $\mathbf{v}_j$ in `FM` because it would be easier to learn the structures by imposing structured sparsity-inducing norm regularization, otherwise we would need to minimize the approximation error to push $\alpha_{ij} \approx \langle \mathbf{v}_i, \mathbf{v}_j \rangle$ for any activated feature pair (i, j). Actually the feature size is relatively small w.r.t. the data size which is usually the case for practical large-scale machine learning problems, thus the complexity of $O(p^2)$ should not be the bottleneck.

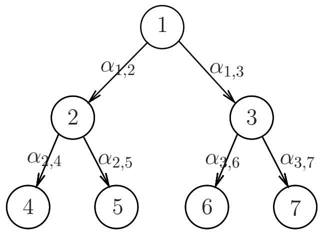

Figure 1: An illustration of hierarchy for features of the same type. Each node corresponds to a feature node, and each edge a feature pair.

2.1 Structured Sparse Priors with Heterogeneous Features

As we discussed before, heterogeneous feature types exist for the user-item-event triplet. For users, it is comparatively easy to obtain certain prior structures for some types of features (e.g., geographical locations), while it is not straightforward for other types (e.g., age and gender).

We partition the features into different types (groups). Let T denote the number of feature types for users and $\mathcal{G} = \{g_1, g_2, \ldots, g_T\}$ the non-overlapping grouping of feature indices, i.e., partition of feature indices by feature types. g_t is the set of feature indices for feature type t. Denote p_t the number of features in feature type t, so that we have $g_t = \{\sum_{s=1}^{t-1} p_s + 1, \sum_{s=1}^{t-1} p_s + 2, \ldots, \sum_{s=1}^{t-1} p_s + p_t\}$ and $\sum_{s=1}^{T} p_s = p$. If we differentiate different types of features, the second-order term in Equation 1 can be further expanded as

$$\underbrace{\sum_{t=1}^{T} \sum_{t'=t+1}^{T} \sum_{k \in g_t} \sum_{k' \in g_{t'}} \alpha_{kk'} \mathbf{x}[k] \mathbf{x}[k']}_{\text{Inter-type Edges}} + \underbrace{\sum_{t=1}^{T} \sum_{\substack{k,k' \in g_t \\ k \neq k'}} \alpha_{kk'} \mathbf{x}[k] \mathbf{x}[k']}_{\text{Intra-type Edges}}.$$

As we partition features into different types, edges of the whole pair-wise feature graph are divided into two parts as well: inter-type edges and intra-type edges. Intra-type edges are initialized from prior knowledge if we have prior hierarchies while inter-type edges are learned. Figure 1 gives an example of the hierarchy for features of one type.

We impose a sparse prior on all inter-typed feature edges, whereas in order to learn sparse hierarchical structure on certain intra-typed feature edges (e.g., geo features), we cast the problem into a hierarchical sparse coding problem by using structured sparsity-inducing norms as regularization [11, 6]. The hierarchical sparsity-inducing regularization has a desirable property that if one edge is included then all its ancestor edges should all be included as well. Equivalently, if one edge is removed from the tree, then all its descendant edges should all be removed too. We can also view the edge hierarchy as pre-ordering of edges, i.e., the parent edge has to be pre-ordered before its child edges. For intra-type edges with hierarchy, we use tree-structured set of groups defined as follows [6].

Definition 1. A set of edge groups $\mathcal{G} \triangleq \{g\}_{g\in\mathcal{G}}$ is said to be tree-structured in $\{1,\ldots,|\mathcal{E}|\}$ if $\bigcup_{g\in\mathcal{G}} g = \{1,\ldots,|\mathcal{E}|\}$ and if for all $g,h \in \mathcal{G}, (g \cap h \neq \emptyset) \Rightarrow (g \subseteq h \textit{ or } h \subseteq g)$, where $\mathcal{E}$ is the set of intra hierarchical feature graph edges. For such a set of groups, there exists a (non-unique) total order relation $\preceq$ such that:

$$g \preceq h \Rightarrow \{g \subseteq h \text{ or } g \cap h = \emptyset\}\,.$$

We can also say that the tree-structured set of groups consists of all the subtrees. As we can see from the definition, we can expand this tree-structured set of groups by adding inter edge and self-edge groups, where each inter edge and self-edge forms a singleton edge group, without affecting the total order relation since they are non-overlapping sets of singleton groups isolated from the tree-structured set of groups. We thus propose to use a mixed singleton and tree-structured sparsity-inducing norm $\mathcal{T}(\boldsymbol{\alpha})$ imposed on the whole feature graph edge weight vector $\boldsymbol{\alpha}$ which is defined as $\mathcal{T}(\boldsymbol{\alpha}) \triangleq \sum_{g\in\mathcal{G}} \left\|\boldsymbol{\alpha}_{|g}\right\|_2$, where $\boldsymbol{\alpha}_{|g}$ is the vector composed of elements of $\boldsymbol{\alpha}$ for indices in edge group g.

2.2 Optimization Problem

Adding the standard l_2-norm regularization on b and $\mathbf{w}$, we want to minimized the regularized risk on all N events

$$\min \frac{1}{N}\sum_{n=1}^{N}\left\|y_n - \hat{y}_n\right\|^2 + \lambda \sum_{\theta\in\{b,w_i\}} \theta^2 + \nu\mathcal{T}(\alpha)\,, \qquad (2)$$

where λ and ν are positive real number parameters for regularization. b and w_i can be updated by alternating least-squares (ALS) optimization by

$$\theta^* = \frac{\theta \sum\limits_{n=1}^{N} h_\theta^2(\mathbf{x}_n) + \sum\limits_{n=1}^{N} h_\theta(\mathbf{x}_n)\, e_n}{\sum\limits_{n=1}^{N} h_\theta^2(\mathbf{x}_n) + N\lambda}, \qquad (3)$$

where $e_n = y_n - \hat{y}_n$ is the training error of the nth event, and

$$h_\theta(\mathbf{x}) = \frac{\partial \hat{y}(\mathbf{x})}{\partial \theta} = \begin{cases} 1, & \text{if } \theta \text{ is } b \\ x[i], & \text{if } \theta \text{ is } w_i \end{cases}.$$

We apply proximal gradient descent to update $\alpha_{|g}$ for each group g in a block coordinate descent strategy where the key step is to compute the proximal mapping w.r.t. l_2-norm

$$\alpha_{|g}{}^{k} = \operatorname{Prox}_{\frac{N\nu}{L}\|\cdot\|_2}\left(\alpha_{|g}{}^{k-1} - \frac{1}{L}\nabla_{\alpha_{|g}} f\left(\alpha_{|g}{}^{k-1}\right)\right),$$

where $f\left(\alpha_{|g}\right) = \frac{1}{N}\sum_{n=1}^{N}\left\|\hat{y}_n - y_n\right\|^2$, and L is the lipschitz constant for f. To speed up the algorithm, we need to estimate L to avoid line search. By applying Gersgorin theorem, it can be proved that $L = 2\left\|\sum_{n=1}^{N} h_{\alpha_{|g}}(\mathbf{x}_n)\, h_{\alpha_{|g}}(\mathbf{x}_n)^T\right\|_\infty$. Note that e_n should be updated accordingly each time b, $\mathbf{w}$, and $\alpha_{|g}$ are updated to guarantee the correctness of the algorithm. The final algorithm has an $O(Np^2)$ time complexity and an $O(p^2 + N\overline{z})$ space complexity where $\overline{z}$ is the average number of non-zero elements for all event feature vectors.

3. EXPERIMENTS

Dataset: We use Yahoo's homepage stream data as a running example of our experiments. A dataset from a sample of traffic of users from six international sites, including United Kingdom, France, Italy, Germany, Spain and Ireland of Yahoo homepage streams is collected. The dataset contains users' interactions, clicks and views, on articles shown in the stream section from May 1st, 2014 to May 16th, 2014. We group users by time periods and construct sessions of all items a user consumed in a particular time period (e.g., months, day, hour and etc.). Mimicking real settings of online systems, events from May 1st to May 9th are used to build the training set, and the remaining events are used to build the test set. The training set has 4.75M events based on the interaction between between $37,668$ users and $30,260$ news items. The test set has 4.46M events based on the interaction between between $45,422$ users and $25,569$ news items.

Features: The critical part of our proposed framework, with other feature-based `CF` is to utilize different types of user and item features. Therefore, in addition to the user interaction data, the dataset also includes the user-side features and item-side features. The user-side feature set consists of three components:

- **Gender**: Categorical, no prior structure, three dimensional (e.g., male, female and unknown).
- **Age**: Categorical, no prior structure, ten dimensional age groups.
- **Geographical areas**: Categorical, a tree-like prior structure, including countries, states and cities. which yields 6744-dimension user-side features. Note that not all features have prior structures. The item-side feature set consists:
- **Content publisher**: Categorical, no prior structure.
- **Topical categories**: Multi-valued (e.g., one article could fall into multiple categories), a directed acyclic graph (DAG) prior structure.

which results in 1347-dimension item-side features. Finally, each event is associated with one item-specific feature denoted as **Freshness**, which measures the lifetime of the item. Summarizing, the dataset has $8,092$-dimension explicit features for all events, the average number of non-zero features for each event is 8.682.

Evaluation Method and Metrics: In addition to evaluating how well we can fit the data by measuring Rooted-Mean-Squared-Error (`RMSE`), we also evaluate the proposed methods in terms of ranking metrics. We use Mean-Average-Precision (`MAP`), Mean-Reciprocal-Rank (`MRR`) and Precision@10 (`P@10`) as main evaluation metrics.

Baselines: We compare our model, denoted as `STSR` with three standard baseline methods:

- `ZeroMean`: A model uses zero vectors as the as priors for user/item latent factors. This includes all classic `LFM` models like [7].
- `RLFM`: A model uses linear regression output as priors for user/item latent factors. This includes models like [1, 3].
- `FM`: A model directly maps features to ratings by a

Table 1: Average performance with standard deviation shown in parentheses. (* represents '0.000', i.e., *14 = 0.00014).

Overall performance

Method	RMSE	MAP	MRR	P@10
`ZeroMean`	.757(*00)	.122(*16)	.149(*20)	.038(*04)
`RLFM`	.339(*07)	.157(*74)	.196(*99)	.052(*31)
`FM`	.329(*00)	.167(*00)	.207(*00)	.053(*00)
`STSR`	.329(*00)	.175(*14)	.218(*15)	.058(*05)

Cold-start performance

Method	RMSE	MAP	MRR	P@10
`ZeroMean`	.961(*13)	.129(*03)	.147(*03)	.037(*02)
`RLFM`	.339(*03)	.166(*73)	.194(*83)	.050(*23)
`FM`	.329(*00)	.176(*00)	.205(*00)	.052(*00)
`STSR`	.329(*00)	.185(*18)	.218(*22)	.057(*06)

second-order regression model with latent feature factorization [9].

Setting: We do grid search for all parameters of all models and report the best average performance in 5 runs. For `ZeroMean` and `RLFM`, we do grid search for σ^2, σ_u^2, and σ_v^2 in $\{10^{-2}, 10^{-1}, \ldots, 10^2\}$, and latent space dimensionality K in $\{5, 10, 20, 50\}$. For `FM`, we do grid search for the regularization parameter λ in $\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$ and latent space dimensionality K in $\{4, 8, 16, 32\}$. For `STSR`, we search λ in $\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$ and ν in $\{10^{-9}, 10^{-7}, 10^{-5}, 10^{-3}, 10^{-1}\}$.

3.1 Experimental Results

We run each model for 5 times from different initialization and average the metrics from multiple runs. The comparison result is shown in Table 1. From the table we see that `STSR` outperforms other models in all ranking metrics significantly. As expected, `ZeroMean` does poorly in all ranking metrics, which validates the statement made in previous studies that it does not handle cold-start problems. `RLFM` does significantly better than `ZeroMean`, implying that features are critical to address cold-start problems and better user/item factors might be learned by utilizing features. `FM` performs significantly better than `RLFM` which is consistent with the statement in literature. Finally, `STSR` learns sparse relationships among features and only the links that are truly contributing to the performance will be enabled through structured sparse coding. Thus, the sparse structure learned from `STSR` turns out making a difference in terms of ranking metrics.

Parameter Analysis: We investigate how different sets of parameters impacting the performance in Figure 2. The optimal parameters are $\lambda = 0.001$, $\nu = 10^{-5}$. The figure shows that `STSR` is not very sensitive to ν with wide ranges, the metrics only differ in ten-thousandths place. `STSR` is mildly sensitive to λ when λ is relatively small, but larger λ does badly hurt the ranking performance. In practice, one can use a validation set (with gold standard) to tune the parameters by grid search.

4. CONCLUSIONS

We proposed a structured sparse regression model that is able to learn heterogeneous sparse structure on the user/item features. We developed an efficient learning algorithm that scales linearly to the number of events. Experiments on a large-scale news recommendation dataset have demonstrated the superior performance of our model compared to the state-of- the-art alternatives, especially for the cold start users, for whom the structure of features turns out to be critical information for the recommendation quality.

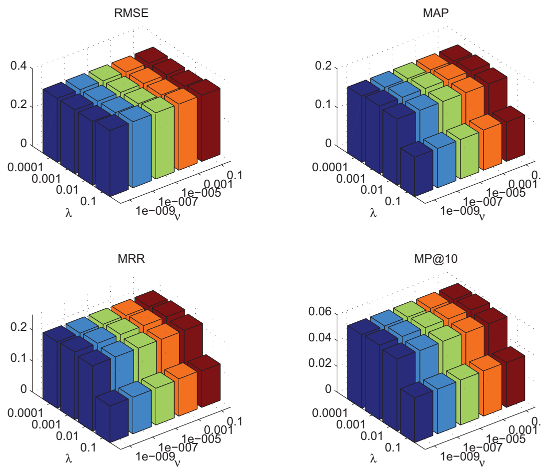

Figure 2: Average metrics in 5 runs with different λ and ν for `STSR`

5. REFERENCES

[1] D. Agarwal and B.-C. Chen. Regression-based latent factor models. In *Proceedings of KDD*, pages 19–28. ACM, 2009.

[2] D. Agarwal and B.-C. Chen. fLDA: matrix factorization through latent Dirichlet allocation. In *Proceedings of WSDM*, pages 91–100. ACM, 2010.

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

[4] Z. Gantner, L. Drumond, C. Freudenthaler, S. Rendle, and L. Schmidt-Thieme. Learning attribute-to-feature mappings for cold-start recommendations. In *Proceedings of the 2010 IEEE International Conference on Data Mining*, ICDM '10, pages 176–185, 2010.

[5] Y. He, Y. Qi, K. Kavukcuoglu, and H. Park. Learning the dependency structure of latent factors. In *Advances in Neural Information Processing Systems 25*, pages 2375–2383. 2012.

[6] R. Jenatton, J. Mairal, G. Obozinski, and F. Bach. Proximal methods for hierarchical sparse coding. *The Journal of Machine Learning Research*, 12:2297–2334, 2011.

[7] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, Aug. 2009.

[8] M. R. Min, X. Ning, C. Cheng, and M. Gerstein. Interpretable sparse high-order boltzmann machines. In *AISTATS*, pages 614–622, 2014.

[9] S. Rendle. Factorization machines with libfm. *ACM Transactions on Intelligent Systems and Technology*, 3(3):57:1–57:22, May 2012.

[10] L. Zhang, D. Agarwal, and B.-C. Chen. Generalizing matrix factorization through flexible regression priors. In *Proceedings of RecSys*, pages 13–20. ACM, 2011.

[11] P. Zhao, G. Rocha, and B. Yu. The composite absolute penalties family for grouped and hierarchical variable selection. *The Annals of Statistics*, pages 3468–3497, 2009.