Place Recommendation with Implicit Spatial Feedback

Berk Kapicioglu (Princeton, Sense Networks), David Rosenberg (Sense Networks),
Robert Schapire (Princeton), Tony Jebara (Columbia)

08/08/2011

1 Introduction

Since the advent of the Netflix Prize [1], there has been an influx of papers on recommender systems in machine learning literature. A popular framework to build such systems has been collobarative filtering (CF) [6]. On the Netflix dataset, CF algorithms were one of the few stand-alone methods shown to have superior performance.

Recently, web services such as Foursquare and Facebook Places started to allow users to share their locations over social networks. This has led to an explosion in the number of virtual places that are available for checking in, inundating users with many irrelevant choices. In turn, there is a pressing need for algorithms that rank nearby places according to the user's interest. In this paper, we tackle this problem by providing a machine learning perspective to personalized place recommendation.

Our contributions are as follows: First, we transform and formalize the publicly available checkin information we scraped from Twitter and Foursquare as a collobarative filtering dataset. Second, we introduce an evaluation framework to compare algorithms. The framework takes into account the limitations of the mobile device interface (i.e. one can only display a few ranked places) and the spatial constraints (i.e. user is only interested in a ranking of nearby venues). Third, we introduce a novel algorithm that exploits implicit feedback provided by users and demonstrate that it outperforms state-of-the-art CF algorithms. Finally, we discuss and report preliminary results on extending our CF algorithm with explicitly computed user, place, and time features.

2 Dataset

Our dataset consists of 106127 publicly available Foursquare check-ins that occurred in New York City over the span of two months. Each datapoint represents an interaction between a user i and a place j, and includes additional information such as the local time of check-in, user's gender and hometown, and place's coordinates and categories. One can view the dataset as a bipartite graph between users and places, where the edges between users and places correspond to check-ins. We preprocess the dataset by computing the largest bipartite subgraph where each user and each place interacts with at least a certain number of nodes in the subgraph. In the end, we obtain 2993 users, 2498 places, and 35283 interactions. We assume that this dataset is a partially observed subset of an unknown binary matrix M of size (m, n), where m is number of users and n is number of places. Each entry $M_{i,j}$ is 1 if user i likes place j, -1 otherwise. Furthermore, denoting the set of all observed incides with Ω, we assume that $\forall\,(i,j) \in \Omega, M_{i,j} = 1$. In other words, if a user checked into a place, we assume she likes the place, but if she didn't check in, we assume that we don't know whether she likes the place or not.

3 Evaluation Framework

Here, we explain the evaluation framework we used to compare algorithms. We assigned each $(i, j) \in \Omega$ to either a train or a test partition. We allowed our algorithms to train on the train partition. Then, for each datapoint (i, j) in the test partition, we only show the algorithms the user i, the place j, and places $N(j)$, where $N(j)$ is the set of neighboring places of j within a given radius (i.e. 500 meters). The algorithms do not know which one of the candidate places is the actual checked in place. We constrain the candidates to be within a certain radius, since we expect the user to be only interested in nearby venues. Algorithms provide a personalized ranking over the given places and we measure the $0-1$ error to predict whether the actual checked in place is among the top t places, where $t \in \{1, 2, 3\}$. Due to the limitations of the mobile device interface, we are only interested in the accuracy of the top few rankings. We do randomized 10 fold cross-validation and report the results.

4 Algorithms

The algorithms we compare range from basic baseline algorithms to the state-of-the-art CF approaches. The baseline algorithms are a predictor that ranks randomly and a predictor that ranks the most popular venue with respect to the training

data. We also have matrix completion algorithms, where the objective is $\min \|A\|_* \; s.t. \; A_{i,j} = M_{i,j}, \forall (i,j) \in \Omega$ and $\|\|_*$denotes the trace norm [5]. In this case, we interpret the entries of the approximation matrix A as confidence scores and rank venues accordingly. This method is justified by [2], where they show that as long as certain technical assumptions are satisfied, with high probability, M could be recovered exactly. In case M is not exactly low rank but approximately low rank, we use Optspace [4]. The use of Optspace is motivated by the upper bound associated with $RMSE\left(M, \hat{M}\right)$, where $RMSE$ indicates root mean square error and $\hat{M}$ indicates the approximate matrix constructed by the algorithm.

The problem with these matrix completion algorithms is that they try to minimize $RMSE\left(M, \hat{M}\right)$. However, what we are interested in is an algorithm that simply scores the actual checked in venue higher than all the neighboring venues. In order to construct such an algorithm, we decided to both measure the performance of vanilla Maximum Margin Matrix Factorization (MMMF) and build upon it [7]. Analogous to how support vector machines were extended to deal with structured outputs, we extended MMMF to rank checked in venues higher than nearby venues. Our approach was partially motivated by [3], where users provided implicit feedback when they clicked on a webpage that was ranked low and the algorithm exploited that feedback. Similarly, everytime a user checks in a venue during training, we assume that she preferred that venue over nearby venues. The optimization problem associated with our algorithm is as follows:

$$\min_{U,V} \frac{1}{2\sqrt{mn}} \left(\|U\|^2 + \|V\|^2 \right) + \frac{\lambda}{|\mathcal{K}|} \sum_{(i,j) \in \Omega} \sum_{k \in N(j)} h\left(\left(UV^T\right)_{i,j} - \left(UV^T\right)_{i,k} \right)$$

where $U \in \mathbb{R}^{m*p}$, $V \in \mathbb{R}^{n*p}$ are the user and place factors, $\mathcal{K} = \{(i,j,k) \mid (i,j) \in \Omega, k \in N(j)\}$ is an extended set of indices, and h is the smooth hinge function

$$h(z) = \begin{cases} \frac{1}{2} - z & z \leq 0 \\ \frac{1}{2}(1-z)^2 & 0 \lneq z \lneq 1 \\ 0 & z \geq 1 \end{cases}$$

The objective is convex and smoothly differentiable in U and V seperately, and similar to [9], we use alternating minimization and BMRM [8] to minimize it. We have a fast implementation in Python where we implemented the objective and gradient computations in C using Cython.

We will demonstrate the complete results of our comparison during the poster session. We will also show some preliminary results on extending the algorithm to exploit explicit user, venue, and time features. However, here's a sneak peek comparing some of the algorithms:

Algorithm	Average Accuracy (within top 1)	Average Accuracy (within top 3)
Random	11.02%	29.94%
Popular	30.67%	52.38%
MMMF	33.06%	55.27%
Spatial MMMF	**35.88**%	**59.11**%

References

[1] James Bennett and Stan Lanning. The netflix prize. In *In KDD Cup and Workshop in conjunction with KDD*, August 2007.

[2] Emmanuel Candès and Benjamin Recht. Exact matrix completion via convex optimization. *Foundations of Computational Mathematics*, 9(6):717–772, December 2009.

[3] Thorsten Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '02, pages 133–142, New York, NY, USA, 2002. ACM.

[4] Raghunandan H. Keshavan, Andrea Montanari, and Sewoong Oh. Matrix completion from noisy entries. *J. Mach. Learn. Res.*, 11:2057–2078, August 2010.

[5] Zhouchen Lin, Minming Chen, and Yi Ma. The augmented lagrange multiplier method for exact recovery of corrupted Low-Rank matrices. arXiv, March 2011.

[6] B. Marlin. Collaborative filtering: A machine learning perspective. Master's thesis, University of Toronto, 2004.

[7] Jasson D. M. Rennie and Nathan Srebro. Fast maximum margin matrix factorization for collaborative prediction. In *Proceedings of the 22nd international conference on Machine learning*, ICML '05, pages 713–719, New York, NY, USA, 2005. ACM.

[8] Choon H. Teo, Alex Smola, Vishwanathan, and Quoc V. Le. A scalable modular convex solver for regularized risk minimization. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '07, pages 727–736, New York, NY, USA, 2007. ACM.

[9] Markus Weimer, Alexandros Karatzoglou, Quoc V. Le, and Alex J. Smola. COFI RANK - maximum margin matrix factorization for collaborative ranking. In John C. Platt, Daphne Koller, Yoram Singer, and Sam T. Roweis, editors, *NIPS*. MIT Press, 2007.