1 Paper Title

Matrix Factorization Techniques for Recommender Systems

2 Summary

This paper highlights the use of latent factor models based on Matrix factorization techniques in collaborative filtering based recommender systems. The earlier systems were heavily using neighbourhood models, but this paper develops a model without neighbourhood techniques and achieves an improvement of 9.46 percent on the existing Netflix algorithm. Also, the authors don't use standard SVD, instead develop a regularised model that can take into account temporal dynamics, implicit feedback and confidence levels by modifying the objective function as discussed further.

3 Detailed Analysis

Matrix factorization techniques based on explicit data factorize a matrix of all user ratings for all products. Such a matrix is very sparse since not every user rates every product. While older papers fill in these sparse values with the mean over user or product vectors and then take SVD, this paper takes into account the observed ratings only and uses Matrix Factorization to model this as a regularized optimization problem, which proves to be a better way than using standard SVD of treating the issue since it doesn't get distorted by the filled-in values.

Let each product i be represented by a vector $q_i \in R^f$ and each user u be represented by a vector $p_u \in R^f$. The dot product $q_i^T p_u$ is the approximate rating of the item i by the user u under this model. Let r_{ui} be the actual rating. Hence the optimization problem becomes:

$$\min_{q*,p*} \sum_{(u,i)\in K} (r_{ui} - q_i^T p_u)^2 + \lambda(\|q_i\|^2 + \|p_u\|^2) \tag{1}$$

The major contribution lies in what happens next. The authors introduce user and product biases by adding bias terms (b_i, b_u), implicit user feedback by adding user preferences for items (item factors $x_i \in R^f$) and user boolean attributes (attribute factor vector $y_a \in R^f$) and show how this model can easily incoporate these techniques.
The opimization problem now becomes:

$$\min_{q*,p*,b*} \sum_{(u,i)\in K} (r_{ui} - \mu - b_i - b_u - q_i^T[p_u + |N(u)^{-0.5} \sum_{i\in N(u)} x_i + \sum_{a\in A(u)} y_a])^2 + \lambda(\|q_i\|^2 + \|p_u\|^2 + b_u^2 + b_i^2) \tag{2}$$

To add temporal data for biases and user preferences, the authors replace the biases with $b_i(t)$ and $b_u(t)$, and the user vector with $p_u(t)$. As a consequence, the rating also becomes a function of time given by $r_{ui}(t)$
Now, to solve this optimization problem and get the latent factor vectors of user and product (p_u and q_i), two techniques are used -

1. Stochastic Gradient Descent: Simple and fast technique, but probably not the best as there are a large number of parameters to optimize.

2. Alternating Least Squares: Fixing either q_i or p_u gives a regression problem for which optimal point is known. Also, easily parallelizable and a good choise when implicit data is also involved.

The authors present their results in terms of the Root Mean Square Error of the model. The model gets better as more and more techniques like biases, temporal dynamics and more data from implicit feedback is taken into account and the RMSE decreases.
Another key implication is that the model doesn't overfit even when parameters increase, and the model performance actually goes up as RMSE goes lower. Also, the model is able to reliably handle huge data.

4 Your critiques

4.1 Assumptions made

One of the assumptions is the authors stating items don't upgrade over time. Let's say we have a new version of a previous item which the seller wants to show as the same as previous item, but an improved version, then this assumption fails.
Another assumption lies in modeling the implicit data. The authors assume the veracity of the implicit data which is harder to verify. What if the user was actually asleep while watching all the episodes of a season. There is no technique for data validation. One suggestion is to use neighbourhood models to find implicit patterns among users.

4.2 Pros and cons

One of the cons is that this model is not very explainable since it uses latent factor vectors which might or might not make sense so as to what each dimension means. This has two implications. First, it becomes a difficult task to verify what the model is learning is correct or not. Second, it becomes difficult to explain to users why they are being recommended a paticular product which degrades the user experience.
Also, relying completely on latent factor models fails to detect localized relationships because such models are focused on overall structure. [1]
Another con is when we introduce another training data point, we have to train the model again as our ratings r_{ui} change and we have no way to know which vectors will change as a result. So if we want updated vectors, we will have to train again.
Another aspect they don't really talk about or focus on is completely new users who don't have any data - explicit or implicit. Or users that have very very less data. How does the algorithm handle that? Perhaps a prior over items is the way to do like in Probabilistic Matrix Factorization[2].
Also what is not discussed and experimented is which models perform better when less ratings are available vs when more ratings are available.
On the other hand, this algorithm is not only memory-efficient, but it is also able to incorporate implicit data, temporal dynamics and confidence levels. Incorporating these subtle data points of implicit feedback is a big reason for their RMSE to be so low and is a major contribution of this work.

4.3 Possible future extensions

One extension is to combine neighbourhood models and latent factor models for explainability as neighbourhood models are more explainable by finding top-k neighbours.
Also, Ensemble models are generally very useful and were responsible for highest improvement of 10.09 percent in 2009. Incorporating Ensemble models with this technique would be a future extension of this.
In the end, to deal with the problem of cold start, Probabilistic and neighbourhood models can be incorporated.

5 Closing Remarks

5.1 What you have learned from reading this paper ?

1. I learned the importance of modeling a problem correctly to solve it efficiently and the effectiveness of matrix factorization techniques that can be used to find relationships between entities.

2. I learned the various algorithms used in the Netflix prize and how they differed and what were the pros and cons of each. (from researching more about this)

3. I also learned the Alternating Least Squares method and found it a bit similar to the EM algorithm we use for Gaussian Mixture Models.

References

[1] KOREN, Y. Factorization meets the neighborhood. *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD 08* (2008).

[2] MNIH, A., AND SALAKHUTDINOV, R. R. Probabilistic matrix factorization. In *Advances in Neural Information Processing Systems 20*, J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, Eds. Curran Associates, Inc., 2008, pp. 1257–1264.

[1] [2]