Lessons from the Netflix Prize Challenge

Robert M. Bell and Yehuda Koren
AT&T Labs – Research
180 Park Ave, Florham Park, NJ
{rbell,yehuda}@research.att.com

ABSTRACT

This article outlines the overall strategy and summarizes a few key innovations of the team that won the first Netflix progress prize.

1. INTRODUCTION

In October 2006, Netflix Inc. released more than 100 million customer generated movie ratings as part of the Netflix Prize competition. The goal of the competition is to produce a 10 percent reduction in the root mean squared error (RMSE) of test data, relative to the RMSE achieved by Cinematch, the technology that currently produces movie recommendations for Netflix customers. A prize of $1,000,000 will be awarded to the first team to reach that goal [5]. These data and the prize have generated unprecedented interest and advancement in the field of collaborative filtering, a class of methods that analyze past user behavior to infer relationships among items and to inform item recommendations for users. Many of these advances have been shared, most notably in the Netflix Prize forum [8] and a 2007 KDD workshop [1].

Three characteristics of the Netflix data combine to pose a large challenge for prediction. First and most obvious is size. Netflix published a comprehensive data set including more than 100 million movie ratings that were performed by about 480,000 users on 17,770 movies. This places a premium on methods that can be computed efficiently. Second, almost 99% of the potential user-item pairs have no rating. Consequently, machine learning methods designed for complete data situations, or nearly so, must be modified or abandoned. Third, the pattern of observed data is very nonrandom; i.e., the amount of observed data varies by more than three orders of magnitude among users or among items. This fact complicates the challenge to detect weak signals for users/movies with sufficient sample size while avoiding over fitting for users/movies with very few ratings.

To further complicate matters, peoples' taste for movies is a very complex process. A particular user's rating of a movie might be affected by any of countless factors ranging from the general—e.g., genre—to quite specific attributes such as the actors, the setting, and the style of background music. Simultaneously modeling such a wide spectrum of factors adds to the challenge.

While no team had reached the 10 percent improvement level after one year, Netflix awarded the first annual progress prize to the KorBell team of AT&T Labs-Research, which achieved the greatest improvement at that time, 8.43%. This article describes some of the strategies that contributed to that team's success.

First, it was important to utilize a variety of models that complement the shortcomings of each other. In particular, this includes both nearest neighbor models (k-NN) and latent factor models such as SVD/factorization or restricted Boltzmann machines. In addition, it was important to include models that incorporated information beyond the ratings themselves—e.g., *what* movies a particular user rated.

Second, we developed several innovations that improved existing collaborative filtering methods, notably:

- A new method for computing nearest neighbor interpolation weights that better accounts for interactions among neighbors.
- A neighborhood-aware factorization method that improves standard factorization models by optimizing criteria more specific to the targets of specific predictions.
- Integration of information about which movies a user rated into latent factor models for the ratings themselves, adapting techniques from [9; 10].
- New regularization methods across a variety of models, including both neighborhood and latent factor models.

Section 2 discusses the need to utilize multiple, complementary models. Section 3 summarizes a few of the innovations developed in the process.

2. UTILIZING A COMPLEMENTARY SET OF MODELS

We found no perfect model. Instead, our best results came from combining predictions of models that complemented each other. While our winning entry, a linear combination of many prediction sets, achieved an improvement over Cinematch of 8.43%, the best single set of predictions reached only 6.57%. Even that method was a hybrid based on applying neighborhood methods to results of a restricted Boltzmann machine. The best improvement achieved purely by a latent factor model was 5.10% and was even less for the best pure nearest neighbor method.

We found that it was critically important to utilize a variety of methods because the two main tools for collaborative filtering—neighborhood models and latent factor models—address quite different levels of structure in the data.

Neighborhood models (k-NN)—the most common form of collaborative filtering—are most effective at detecting very

localized relationships. The item neighborhood approach identifies pairs of items (movies) that tend to be rated similarly, in order to predict ratings for an unrated item based on ratings of similar (neighboring) items by the same user [11]. Similarly, the user approach bases predictions on ratings of the same item by similar users [7]; however, the rest of this article considers only the item approach. Our best neighborhood models typically used 20 to 50 neighboring movies for any single prediction, often ignoring the vast majority of ratings by the user of interest [2]. Consequently, these methods are unable to capture the totality of weak signals encompassed in all of a user's ratings.

Latent factor models comprise an alternative approach to collaborative filtering with the more holistic goal to uncover latent features that explain the observed ratings. For an $m \times n$ ratings matrix R with no missing values, matrix factorization via singular value decomposition (SVD) approximates R with the best rank-f approximation R^f, defined as the product of two rank-f matrices $P_{m\times f}$ and $Q_{n\times f}$, where f is usually much smaller than either m or n. The matrix R^f captures the f most prominent features of the data, leaving out less significant patterns in the observed data that might be mere noise. For sparse data like the Netflix ratings, alternative estimation methods are required in order to deal with the missing elements and to avoid over fitting (see, e.g., [3; 4; 6; 10]). Restricted Boltzmann machines provide another model based on estimating latent factors [10].

Latent factor models are generally effective at estimating overall structure that relates simultaneously to most or all movies. However, these models are poor at detecting strong associations among a small set of closely related movies such as The Lord of the Rings trilogy, precisely where neighborhood models do best.

While, in theory, either methodology might be extended to encompass the full spectrum of relationships, such an effort seems doomed to failure in practice. Finding niche clusters of items such as Hitchcock films or boxing movies might require thousands of latent factors. And even so, those factors might be swamped by the noise associated with the thousands of other movies. Similarly, for neighborhood methods to capture the degree that a user prefers action to dialogue may require a very large number of neighbors, to the exclusion of estimating the user's interest in other relevant features.

We also found it extremely value to utilize models that incorporate information beyond simply the ratings themselves. This was important for better modeling the users, as most users in the test set provided a meager number of ratings. In pursuing this approach, we analyzed *which* movies users rate, regardless of *how* they rated these movies. Hence, we try modeling for which movie a user will choose to voice his/her opinion and vote a (positive or negative) rating. This provided an important tool for learning about users, generating a unique perspective that nicely complemented the ratings-based information. We call this "the binary view of the data". Overall, this was a key to our progress, and will probably serve an even bigger role in real life recommender systems, where richer and comprehensive implicit user behavior (e.g., rental history, browsing history and search patterns) is available to enrich models centered around explicit ratings.

Finally, some of our models utilized other information about items, users, or individual ratings such as the number of ratings of an item or by a user, the average rating of an item or by a user, and the date of the rating (see section on "Global Effects" in [2] for details). Variables like these allowed us, for example, to distinguish users who like the most commonly rated movies best from those who prefer more specialized fare, or to predict better for users whose ratings rise over time, above and beyond any change explained by the inherent quality of the items being rated.

3. IMPROVING EXISTING METHODS

Generally speaking, improved modeling tends to follow one of three directions, or a combination thereof:

1. Deepening known methods in order to improve their quality
2. Combining multi-scale views of the data
3. Combining explicit rating information with implicit rating behavior (the binary view)

Accordingly, we will demonstrate our improvements on two well known models.

3.1 Weights for Neighborhood Models

Our goal is to predict an unobserved rating by user u for item (movie) i, denoted as r_{ui}. An item-oriented neighborhood model identifies a set of neighboring *items* $\mathrm{N}(i;u)$ that other users tend to rate similarly to their rating of i. All items in $\mathrm{N}(i;u)$ must have been rated by u. The predicted value of r_{ui} is taken as a weighted average of the ratings of neighboring items:

$$\hat{r}_{ui} \leftarrow b_{ui} + \frac{\sum_{j\in \mathrm{N}(i;u)} s_{ij}(r_{uj} - b_{uj})}{\sum_{j\in \mathrm{N}(i;u)} s_{ij}} \qquad (1)$$

The item similarities—denoted by s_{ij}—play a central role here, as they are used both for selecting the neighbors and for weighting the above average. Common choices are the Pearson correlation coefficient and the closely related cosine similarity. We use b_{ui} to denote some baseline predictor for r_{ui}. It is important to use these baseline values to remove item- and user-specific biases that may prevent the model from revealing the more fundamental relationships. Prior methods often take b_{ui} as the mean rating of user u or item i. In [2; 3] we offered a more comprehensive approach to these baseline estimates.

Neighborhood-based methods became very popular because they are intuitive and relatively simple to implement. In particular, they do not require tuning many parameters or an extensive training stage. They also provide a concise and intuitive justification for the computed predictions. However, standard neighborhood-based methods raise some concerns:

1. The similarity function s_{ij}, which directly defines the interpolation weights, is arbitrary. Various algorithms use somewhat different similarity measures, trying to quantify the elusive notion of item similarity. Suppose that a particular item is predicted perfectly by a subset of the neighbors. In that case, we would want the predictive subset to receive all the weight, but that is impossible for bounded similarity scores like the Pearson correlation coefficient.

2. Previous neighborhood-based methods do not account for interactions among neighbors. Each similarity between an item i and a neighbor $j \in \mathrm{N}(i;u)$ is computed independently of the content of $\mathrm{N}(i;u)$ and the other similarities: s_{ik} for $k \in \mathrm{N}(i;u) - \{j\}$. For example, suppose that our items are movies, and the neighbors set contains three movies that are highly correlated with each other (e.g., sequels such as "Lord of the Rings 1–3"). An algorithm that ignores the similarity of the three movies when determining their interpolation weights, may end up essentially triple counting the information provided by the group.
3. By definition, the interpolation weights sum to one, which may cause overfitting. Suppose that an item has no useful neighbors rated by a particular user. In that case, it would be best to ignore the neighborhood information, staying with the current baseline estimate. Nevertheless, the standard neighborhood formula uses a weighted average of ratings for the uninformative neighbors.
4. Neighborhood methods may not work well if variability differs substantially among neighbors.

We propose a method that overcomes these difficulties. First, we replace the weighted average by a more general weighted sum, which allows downplaying neighborhood information when lacking informative neighbors. Given a set of neighbors $\mathrm{N}(i;u)$ we need to compute *interpolation weights* $\{w_{ij} | j \in \mathrm{N}(i;u)\}$ that will enable the best prediction rule of the form:

$$\hat{r}_{ui} \leftarrow b_{ui} + \sum_{j \in \mathrm{N}(i;u)} w_{ij}(r_{uj} - b_{uj}) \qquad (2)$$

We learn interpolation weights by modeling the relationships between item i and its neighbors through a least squares problem:

$$\min_{w} \sum_{v \neq u} \left(r_{vi} - b_{vi} - \sum_{j \in \mathrm{N}(i;u)} w_{ij}(r_{vj} - b_{vj}) \right)^2 \qquad (3)$$

The major challenge is to cope with the missing values, and this can be done efficiently by estimating all inner products between movie ratings. For a full description refer to [3]. Importantly, this scheme provides interpolation weights that are derived directly from the ratings residuals $(r_{uj} - b_{uj})$, not based on any similarity measure. Moreover, derivation of the interpolation weights explicitly accounts for relationships among the neighbors.

Our experiments showed that this scheme significantly improved accuracy relative to that based on more standard interpolation weights (e.g., driving RMSE from around 0.94 to around 0.92 for Netflix's probe data) without a meaningful increase of running time [3].

So far, we demonstrated the benefits of building a deeper, fundamental model for k-NN. Additional accuracy gains are possible by combining the local scale view of k-NN with the higher scale view of latent factor model. Accordingly, we can take the baseline estimates (the b_{ui}'s) as the predictions made by a latent factors model, what amounts to activating k-NN on the residuals of factorization. This enabled us to achieve RMSE of 0.898 for the test data, benefiting from the multi-scale view of the data. Further improvement is achieved by integrating also the "binary viewpoint". One way to achieve this is by utilizing conditional restricted Boltzmann machines (RBM) [10]. The RBM model achieves a similar accuracy to standard factorization model, while relying also on the binary viewpoint. When applying k-NN on residuals of RBM, thereby integrating all important viewpoints of the data (local, high scale and binary), RMSE dropped to 0.889.

3.2 Latent Factor Models

Latent factor models measure the agreement of users and movies across a series of features that are algorithmically learned from the data. This way, we associate each user u with a user-factors vector $p_u \in \mathbb{R}^f$, and each movie with a movie-factors vector $q_i \in \mathbb{R}^f$. The prediction is done by taking an inner product – $\hat{r}_{ui} = p_u^T q_i$. The more involved part is the estimation of the factors. A straightforward approach would minimize the regularized cost function:

$$\sum_{(u,i) \in \mathcal{K}} (r_{ui} - p_u^T q_i)^2 + \lambda(||p_u||^2 + ||q_i||^2) \qquad (4)$$

The set $\mathcal{K}$ contains all (u,i)-pairs for which r_{ui} is known. The regularization parameter λ prevents overfitting; a typical value is $\lambda = 0.05$. We minimized (4) by an alternating least squares scheme (Sec. 5.1 of [3]). Others [6; 10] used gradient based techniques. When using 60 factors ($f = 60$), this basic factor model predicts the test set with RMSE 0.908.

Interestingly, this basic factorization model can benefit from moving along each of the three possible directions outlined in the beginning of the section. First, we can improve accuracy by deepening the foundation of the model. The squared penalty used in (4) implicitly assumes that all factors are drawn from independent normal distributions with zero mean and the same variance. This is a quite limiting assumption. A richer model assumes a general multivariate normal distribution for the factors. Accordingly, the ratings can be modeled by the following. First, we draw user and movie factors from the joint normal distribution:

$$p_u \sim N(\mu, \Sigma) \qquad q_i \sim N(\gamma, \Lambda) \qquad (5)$$

Then, each rating r_{ui} is taken from the normal distribution:

$$r_{ui} = N(p_u^T q_i, \epsilon)$$

The priors given in (5) avoid overfitting by shrinking the factors towards baseline values when not enough ratings are available. Removing the parameter restrictions implied by (4) produces significant accuracy gains. For a 60 factors example, the RMSE on the test data fell to 0.899.

Naturally the added expressive power of a richer Gaussian prior comes with an added complexity of the optimization algorithms. We employed two techniques. One is based on Gibbs sampling, and the other is based on expectation maximization, where we alternate between fixing the factors and fixing the parameters; see [12] for a related technique.

Alternatively, we can stay within the convenient simple formulation (4) and significantly improve prediction accuracy by integrating either the localized viewpoint or the binary viewpoint into the model. Integrating the local viewpoint gives us a more complete multi-scale view of the data. To this end we use *adaptive user factors* that model the behavior of a user within the neighborhood of the given movie; see [4]. Technically, we allow user factors to change according to

the item that is being predicted. Therefore, when predicting r_{ui} we first compute a vector $p_u(i) \in \mathbb{R}^f$ – depending on both u and i – and then predict r_{ui} as $(p_u(i))^T q_i$. The computation of $p_u(i)$ is done by tilting the squared error (4) to overweight those movies similar to i, obtaining the problem:

$$\min_{p_u(i)} \sum_{j \in \mathrm{N}(u)} s_{ij}(r_{uj} - p_u(i)^T q_j)^2 + \lambda ||p_u(i)||^2 \qquad (6)$$

Here, $\mathrm{N}(u)$ is the set of movies rated by user u. The constant s_{ij} is a similarity measure between movies i and j. By integrating the local viewpoint, prediction accuracy is significantly improved. For the 60 factors case, the error drops to RMSE=0.897.

Instead of integrating the local viewpoint, we can integrate the binary viewpoint. Here we adopt a principle from Paterek's NSVD method [9]. The NSVD model refrains from explicitly parameterizing each user, but rather models each user based on the movies that he/she rated. This way, each movie i is associated with two movie-factors vectors q_i and x_i. The representation of a user u is through the weighted sum: $\left(\sum_{j \in \mathrm{N}(u)} x_j\right) / \sqrt{|\mathrm{N}(u)|}$, so r_{ui} is predicted as: $q_i^T \left(\sum_{j \in \mathrm{N}(u)} x_j\right) / \sqrt{|\mathrm{N}(u)|}$. Importantly, NSVD models user behavior purely based on the binary viewpoint. This allows a straightforward integration of the binary viewpoint into the basic factorization model. A rating is modeled by

$$\hat{r}_{ui} = q_i^T \left(p_u + \left(\sum_{j \in \mathrm{N}(u)} x_j \right) / \sqrt{|\mathrm{N}(u)|} \right) \qquad (7)$$

The parameters are estimated by gradient descent minimizing of the associated regularized squared error. The model used with 60 factors predicts the test set with RMSE lower than 0.897.

To summarize, the accuracy of the latent factors model can be improved by following three distinct directions. First, one can deepen the foundations of the model, by assuming a more flexible and realistic probabilistic model. Alternatively, one can retain the simple structure of the original model, but gain even more accuracy by introducing complementing perspectives of the data into the model, being either the local view or the binary view.

3.3 Regularization

Regularization plays a central role in all of our methods. All the models described above include massive numbers of parameters, many of which must be estimated based on a small number of ratings (or pairs of ratings). Consequently, it is critically important to avoid over fitting the training data. We use shrinkage to improve the accuracy of similarity scores used for nearest neighbor selection [4]. Shrinkage of estimated inner products is essential to successful estimation of interpolation weights for the nearest neighbor model described above [2]. Our factorization models typically use ridge regression for regularization. Removal of global effects used empirical Bayes shrinkage [2].

4. DISCUSSION

Our experience with the Netflix competition showed that the most successful model is an ensemble of multiple predictors; each of them specializes in addressing a different aspect of the data. In particular, most improvement stems from moving models along three different axes. The first (and most obvious) axis denotes improving the quality of the models by basing them on deeper foundations. The second axis spans a multi-scale modeling of the data that integrates the localized perspectives expressed within limited neighborhoods with the more regional view expressed by latent factors models. The third axis introduces the "binary view" of user behavior that models which movies the users are likely to rate regardless of the value of the rating. This binary view can be explored by working with a compatible binary representation of the data or, alternatively, by employing models that allow expressing binary aspects within the context of ratings data. These models include conditional restricted Boltzmann machines[1][10], NSVD [9], and some related specialized models.

Although the winning Netflix entry combined many sets of predictions, we note that it is possible to achieve a 7.58% improvement with a linear combination of three prediction sets that combine all the key elements discussed in this article: k-NN models, removal of global effects, neighborhood aware factorization, and latent factor models with the binary viewpoint.

Real life recommender systems are exposed to two types of user feedback. One is the high quality explicit rating, where users clearly express their opinion on a product. However, this kind of explicit feedback may be difficult to collect due to system constraints and reluctance of users to cooperate. A second source of information is the abundant implicit feedback, which may be purchase/rental history, browsing patterns, search keywords, etc. Typically, a recommender system capitalizes on only one of these two sources of information. We believe that the "binary view" serves as a proxy for general implicit feedback, suggesting that a combined system utilizing both explicit and implicit input is most desirable. While the Netflix data allowed us to use only the "who rated what" information, a real life system can exploit the much broader "who rented what" information, together with all other sorts of implicit feedback. This implies that our very positive experience of integrating the implicit binary view into our models can be leveraged by real life systems that access wide ranging implicit feedback. The most challenging part of the Netflix data proved to be the many users who provided very few ratings. It is very likely that many of those users do have a rich rental history, which could allow a combined recommender system to make accurate, personalized recommendations for them, or even for customers who provided no ratings at all.

5. REFERENCES

[1] ACM SIGKDD, "KDD Cup and Workshop 2007", www.cs.uic.edu/~liub/Netflix-KDD-Cup-2007.html.

[2] R. M. Bell and Y. Koren, "Improved Neighborhood-based Collaborative Filtering", *Proc. KDD-Cup and Workshop at the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2007.

[1]While conditional RBMs provide a tool for exploiting additional information within Netflix's test set [10], we believe that their real strength is efficient integration of the implicit binary view within the explicit ratings. In our tests, most improvement over non-conditional RBMs was realized without including the implicit information in Netflix's test set.

[3] R. M. Bell and Y. Koren, "Scalable Collaborative Filtering with Jointly Derived Neighborhood Interpolation Weights", *Proc. IEEE International Conference on Data Mining (ICDM'07)*, 2007.

[4] R. M. Bell, Y. Koren and C. Volinsky, "Modeling Relationships at Multiple Scales to Improve Accuracy of Large Recommender Systems", *Proc. 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2007.

[5] J. Bennett and S. Lanning, "The Netflix Prize", *Proc. KDD-Cup and Workshop at the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2007.

[6] S. Funk, "Netflix Update: Try This At Home", http://sifter.org/~simon/journal/20061211.html, 2006.

[7] J. L. Herlocker, J. A. Konstan, A. Borchers and J. Riedl, "An Algorithmic Framework for Performing Collaborative Filtering", *Proc. 22nd ACM SIGIR Conference on Information Retrieval*, pp. 230–237, 1999.

[8] Netflix, Inc., "Netflix Prize: Forum", www.netflixprize.com//community/.

[9] A. Paterek, "Improving Regularized Singular Value Decomposition for Collaborative Filtering", *Proc. KDD-Cup and Workshop at the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2007.

[10] R. Salakhutdinov, A. Mnih and G. Hinton, "Restricted Boltzmann Machines for Collaborative Filtering", *Proc. 24th Annual International Conference on Machine Learning (ICML'07)*, 2007.

[11] B. Sarwar, G. Karypis, J. Konstan and J. Riedl, "Item-based Collaborative Filtering Recommendation Algorithms", *Proc. 10th International Conference on the World Wide Web*, pp. 285-295, 2001.

[12] Y. Zhang and J. Koren, "Efficient Bayesian Hierarchical User Modeling for Recommendation Systems", *Proc. 30th International ACM Conference on Research and Development in Information Retrieval (SIGIR'07)*, 2007.