Matrix Factorization for Collaborative Filtering

Calvin Roth

December 19, 2021

Contents

1 Abstract

Big tech companies including Amazon, Youtube, and Netflix use recommendation systems as a center piece of their business models and many other companies are seeing the use of implementing such systems. The question of how to best apply optimization techniques to give good recommendations in scalable way is an active area of research. In this paper, we will look at some of the recent developments in collaborative filtering.

2 Introduction

Recommendation systems are a long application area at the heart of many tech companies and research. A sizable fraction of major websites on the Internet rely on being able to provide good recommendations for people ranging from what to buy for basically every store and what to watch/listen to next for media companies like Youtube or Tiktok. ACM to meet these needs as a conference just for Recommendation systems research and IEEE and others happily welcome new Recommendation system algorithms.

Collaborative filtering is the idea that we should use the similarities between other user-item interactions to inform our guesses for recommendations. A basic example is since people who like the artist you are currently listening to like this other artist maybe you would too. The history of this term comes from one of the first systems, Tapestry, which had the goal making a system where users would give their reactions to items and this collective action would help lead to good ways to filter out the boring content from various emailing lists[1] even if today the fact the collaborative word seems like an odd fit.

One variant of collaborative filtering computes the similarity between items or between users by various and find uses a K-nearest neighbors approach to generate new recommendations[2]. For example, for a user-based system we look at the most similar users to one individual and recommend items that his/her neighbors like. The other model, the focus of this paper is called the model based approach[3]. These algorithms aim to learn a set of matrices that represent a lower dimensional space and how much weight to give each element of this lower dimensional space.

The starting framework for model based collaborative filtering is we are given a matrix A of where the rows are users and the columns are items. We want to learn two matrices W, H such that $A \approx WH$ but with the added restriction that each row of W and each column of H live the lower dimensional rather then the full dimensional space of A which is a priori not obvious what it should be as a general rule.

The standard objective function is $\min_{W,H} \|A - WH\|_F^2$ because we want W*H to be close to A entry wise. If it were the case that A was fully known then this easy. We can just take $[U, S, V] = svd(A)$ and assign $W = U_k S_k^{\frac{1}{2}}$ and $H = S_k^{\frac{1}{2}} V_k$ where is the size of reduced space[3]. There a many ways to add on to this model with the hope of in addition creating a sparse matrix or

preventing overfitting for instance. Below I will discuss some of the methods that have seen success and will implement some of them.

3 Methods

3.1 PureSVD

If our goal is to minimize $\|A - XY\|_F^2$ then a very natural start is to pick X,Y based on the SVD of A. The reason why this isn't the end of the story is A has many missing reviews. If A were completely known then $X = U\Sigma^{\frac{1}{2}}$ and $Y = \Sigma^{1/2}V'$ where U, Σ, V is the svd of A. Let $[U_k, \Sigma_k, V_k']$ by the truncated SVD of A. The pureSVD[4] uses treats the missing values as 0 and otherwise entries by their raw score. This is different than other methods we will see assume the input matrix is binary with 1 for have purchased and 0 for has not yet purchased. The method proceeds by setting $X = U_k\Sigma_k$ and $Y' = V_k'$. From this the recommendations for user i are the indices of the k largest elements from the ith row of XY. When testing this method after computing X and Y, I projected the entries of $X * Y$ to the range 0-5 as that is the range of possible review scores so doing so can only decrease the loss. On the other hand there are no restrictions placed on the entries of X and Y.

3.2 Maximum Margin

The idea behind this method is akin to SVM methods. The input data will be observed labels for user-item pairs where an it is marked as $+1$ is the user liked and -1 if they did not. Following the paper[5], we will call Y the matrix of all user-item pairs, S the set of each of rated user-item pairs, and Y_S the restriction of Y to S. In this method we wish to find a low rank matrix X that approximates Y_S. If X is of rank k and $Y \in \mathbb{R}^{n\times m}$ then X can be factorized in $X = UV, U \in \mathbb{R}^{n\times k}, V \in \mathbb{R}^{k\times m}$. Here we can think of the rows of U as feature vectors and the columns of V as the weights to apply for each user. The authors instead of imposing the constraint in the rank of X add a penalty that punishes high rank matrices. This is to make the problem more tractable. They choose the penalty term $\|X\|_*$ where $\|\cdot\|_*$ is the nuclear norm of X. In addition, they impose X be close to the observed values we add a hinge loss for each element in S. Overall then the optimization problem

proposed is

$$\min_X \|X\|_* + c \sum_{(i,j)\in S} max(0, 1 - Y_{i,j}X_{i,j}) \tag{3.1}$$

This is a convex problem and the authors explained that the set of matrices with bounded spectral norm is a closed convex set. They approach this problem by writing the dual problem of 3.1 and dual variable Q.

$$\begin{aligned} &\max_Q \sum_{(i,j)\in S} Q_{i,j} \\ &\|Q \otimes Y\|_2 \leq 1 \\ &0 \leq Q_{i,j} \leq c \forall (i,j) \in S \end{aligned} \tag{3.2}$$

Where $\otimes$ is the entry wise product of two matrices.

3.3 Alternating Least Squares

This paper[6] and one we will see later both employ the rule that observed behavior is counted as $+1$ and unobserved is 0. In this way we only learn from what what elements are in the class which is why the paper is called one class collaborative filtering, one observations in the purchased class are given weight. Imagine a website like Amazon, it is your purchases that influence future recommendations instead of some combination of items you haven't yet purchased. We will try to create a low rank matrix X that approximates the sample data A. As mentioned above, the standard loss function is $\min_X \|A - X\|_F^2$. For the algoirthm we will actually build the factorization of $X = UV$. To prevent overfitting we will also add penalty term on the size of U and V

$$\min_{U,V} \sum_{i,j} \left(\|A_{ij} - U_i * V_j^T\|_F^2 \right) + \lambda(\|U\|_2^F + \|V\|_F^2) \tag{3.1}$$

Note that the authors include the last two norms as part of the sum instad of outside it.

The optimality condition for U_{ij} is then:

$$2 \sum_k \left((U_i V_k^T - A_{ij}) V_{kj} \right) + \lambda U_{ij} = 0$$

We can write this in vector form for the ith column as

$$U_i(V^T * V + \lambda I) - A_i V = 0$$
$$U_i = (A_i V) * inv(V^T V + \lambda I) \tag{3.2}$$

The case for V is symmetric and condition is

$$V_i = (A_i^T U) inv(U^T V + \lambda * I) \tag{3.3}$$

The alternating least squares ideas is to start with some initial random matrix $U^{(0)}$ and from there build $V^{(0)}$ from 3.3 using this U and then optimize U using this V and just repeatedly alternate between optimizing U with respect to V and then V with respect to U.

In this method it is is not clear what the rank of X should be. There is a trade off between higher rank X which has more expressivity but will take longer to train.

3.4 SLIM

SLIM, or Sparse LInear Method, is an algorithm developed here at the Minnesota which has the name suggests aims to preserve sparsity in the prediction matrix[7]. Here the authors investigated recommendations with the input data being binary for each (user, item) pair in a matrix A. If $A_{ij} = 1$ then user i has purchased item j and 0 otherwise. This is works naturally for an e-commerence site like Amazon but may need adjustments for applications where we care about the rating too. For example, if I watch a movie and hate it, you should not give it positive weight for future recommendations. The easiest fix to make the algorithm presented in this paper work is threshold ratings so that scores above a value are 1 and below are 0.

From this matrix of purchase history, we wanted to make a weight matrix W to give each individual weights to different items. To get a recommendation once we know W for user i, we can just take $A_{i,:}W$ to get a prediction of scores for all the items. From this vector of scores, we take the top k scoring items that have not been purchased by user i and offer them as suggestions. Finally, we require that W be non-negative and that $W[i, i] = 0$ for all i. The first requirement is so the weight matrix only is positive relations between items and the 2nd eliminates a trivial solution of the identity.

This objective function is

$$min_W \frac{1}{2}\|A - A * W\|_F^2 + \frac{\beta}{2}\|W\|_F^2 + \lambda\|vec(W)\|_1 \tag{3.1}$$
$$W_{ij} \geq 0 \forall i, j$$
$$W_{ii} = 0$$

By $\|vec(W)\|_1$ I mean $\sum_{i,j} |W_{ij}|$ i.e. the ℓ_1 norm of W if the entries of W were in a vector. Alternatively you can view this as the ℓ_1 analogue of what the Frobenius norm is for the ℓ_2 norm but notation for what to call this doesn't seem fixed in literature. The first time just enforces that the weight matrix be close to what we observed without being the identity. The second term promotes sparsity in W, with higher β encouraging more sparsity. The second and third term coupled together constitute an elastic net penalty which has been shown to reduced overtraining as well as making more similar users are more likely to observe similar assignments, an effect called grouping[8].

What really makes this model shine is the optimization procedure. Notably this is separable.

$$\|A - A * W\|_F^2 = \sum_i \|A_{:,i} - AW_{:,i}\|_2^2$$
$$\|W\|_F = \sum_i \|W_{:,i}\|_2^2$$
$$\|vec(W)\|_1 = \sum_i \|W_{:,i}\|_1$$

So we can decompose this problem into the problems of solving for each column of W independently.

Now we examine how to solve this optimize this which was detailed by Friedman et. al[9] which gives a coordinate descent technique to optimize w_j.

In order to solve each of these subproblems we first look at the optimality conditions for update $W_{i,j}$ the jth coordinate of W_i:

$$\frac{\partial f}{\partial W_i}\frac{1}{2}\|A_i - AW_i\|_2^2 + \frac{\beta}{2}\|W_i\|_2^2 + \lambda\|W_i\|_1 = 0$$
$$-A_{,j}^T(A_i - AW_i) + \beta W_{i,j} + \lambda sign W_{i,j} = 0$$

The optimal choice of W_{ij} is found by choosing $W_{ij} = \frac{S(A_{,j}(A_i - AW_i^{\odot}),\lambda)}{1+\beta}$

Where $S(x,y)$ is the soft-thresholding function i.e. it is $\begin{cases} x-y & \text{If } |x| > y \\ x+y & \text{If } |x| > -y \\ 0 & \text{Else} \end{cases}$.

This was shown in [10] using techniques from [11].

Finally, to maintain the constraints we project the solution for W_{ij} to $\mathbb{R}^{\geq 0}$.

4 Results

4.1 The Data

We will examine the perform of these methods testing on the Movie lens small data set[12]. This data consists of approximately 100000 ratings for 10000 movies and 610 users. We implemented all the codes besides Maximum margin factorization but only had success in testing with alternating least squares and PURESVD. The rest can be found at or in the ZIP file.

4.2 Tests and Parameters

For Alternating least squares, I set $\lambda = 1$ because it performed well and the orignal paper also didn't do anything clever to pick λ[6]. As for the dimensions of U, V, we first compute the SVD of the training set. Letting T be the sum of the singular values we will take as the inner dimension to be number of singular values whose sum is certain fraction of T. Here, I tested the number of dimensions needed such that the sum of singular values is 70%, 90%, and 99% of total nuclear norm. For our data, 252 singular values covers 70%, 434 for 90%, and 578 for 99%.

Computing these sizes took 13.49 seconds. Below we report the loss for each of these tests where the loss is that of 3.1

As we can see there is a large improvement from going from 252 to 434 as the latent dimension also a severe increase in time needed. When we allow another 140 dimensions we get another big increase in the time needed to converge but a much small boost in accuracy.

For PUREsvd, we used the same dimensions for the size of SVD to take

Size of Latent Dimension	Runtime(in seconds)	Loss
252	259	14820
434	863	10929
578	1193	10290

Table 1: Results for Alternating Least Squares algorithm

Size of Latent Dimension	Runtime(in seconds)	Loss
252	15.58	190530
434	15.78	198355
578	17.83	199406

Table 2: Results for PUREsvd

Here we see that in terms of the time the size to allow U and V to be doesn't make a large difference in terms of time and increasing it surprising increases the loss. The exact values of the loss are difficult to directly compare with the ALS test because here entries are allowed to be in the domain 0-5 instead of 0-1.

References

[1] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry, "Using collaborative filtering to weave an information tapestry," *Commun. ACM*, vol. 35, p. 61–70, dec 1992.

[2] X. Su and T. M. Khoshgoftaar, "A survey of collaborative filtering techniques," *Advances in artificial intelligence*, vol. 2009, 2009.

[3] P. Melville and V. Sindhwani, "Recommender systems.," *Encyclopedia of machine learning*, vol. 1, pp. 829–838, 2010.

[4] P. Cremonesi, Y. Koren, and R. Turrin, "Performance of recommender algorithms on top-n recommendation tasks," in *Proceedings of the fourth ACM conference on Recommender systems*, pp. 39–46, 2010.

[5] N. Srebro, J. D. Rennie, and T. S. Jaakkola, "Maximum-margin matrix factorization.," in *NIPS*, vol. 17, pp. 1329–1336, Citeseer, 2004.

[6] R. Pan, Y. Zhou, B. Cao, N. N. Liu, R. Lukose, M. Scholz, and Q. Yang, "One-class collaborative filtering," in *2008 Eighth IEEE International Conference on Data Mining*, pp. 502–511, IEEE, 2008.

[7] X. Ning and G. Karypis, "Slim: Sparse linear methods for top-n recommender systems," in *2011 IEEE 11th International Conference on Data Mining*, pp. 497–506, IEEE, 2011.

[8] H. Zou and T. Hastie, "Regularization and variable selection via the elastic net," *Journal of the royal statistical society: series B (statistical methodology)*, vol. 67, no. 2, pp. 301–320, 2005.

[9] J. Friedman, T. Hastie, and R. Tibshirani, "Regularization paths for generalized linear models via coordinate descent," *Journal of statistical software*, vol. 33, no. 1, p. 1, 2010.

[10] J. Friedman, T. Hastie, H. Höfling, and R. Tibshirani, "Pathwise coordinate optimization," *The annals of applied statistics*, vol. 1, no. 2, pp. 302–332, 2007.

[11] D. L. Donoho and J. M. Johnstone, "Ideal spatial adaptation by wavelet shrinkage," *biometrika*, vol. 81, no. 3, pp. 425–455, 1994.

[12] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context," *ACM Trans. Interact. Intell. Syst.*, vol. 5, dec 2015.