NONNEGATIVE MATRIX FACTORIZATION UNDER ADVERSARIAL NOISE

Peter Ballen

Department of Computer and Information Science, University of Pennsylvania, Philadelphia, USA

ABSTRACT

Nonnegative Matrix Factorization (NMF) is a popular tool to estimate the missing entries of a dataset under the assumption that the true data has a low-dimensional factorization. One example of such a matrix is found in movie recommendation settings, where NMF corresponds to predicting how a user would rate a movie. Traditional NMF algorithms assume the input data is generated from the underlying representation plus mean-zero independent Gaussian noise. However, this simplistic assumption does not hold in real-world settings that contain more complex or adversarial noise. We provide a new NMF algorithm that is more robust towards these nonstandard noise patterns. Our algorithm outperforms existing algorithms on movie rating datasets, where adversarial noise corresponds to a group of adversarial users attempting to review-bomb a movie.

KEYWORDS

Nonnegative Matrix Factorization, Matrix Completion, Recommendation, Adversarial Noise, Outlier Detection, Linear Model

1. INTRODUCTION

Large datasets are commonplace in multiple data science applications. These datasets suffer from two key properties that makes analysis challenging. The first challenge is that the datasets are high-dimensional, with each item in the dataset containing many features. The well-cited "curse of dimensionality" causes standard machine learning techniques to offer suboptimal performance on these large datasets [1,2]. The second challenge is that the datasets contain large amounts of missing data.

To handle the first problem, matrix factorization is a popular preprocessing step on numeric data to reduce the dimensionality of the data to a more manageable size. The true data is assumed to have a low-dimensional linear model. The user has access to noisy samples from this model and must learn the underlying model from the samples. The user then applies the learned model to predict the missing data. See [3,4,5] and references within.

In mathematical terms, the input data is treated as a sparse n x m matrix **V** with missing entries. The goal to find an n x k matrix **W** and a k x m matrix **H** such that the matrix product **WH** approximates the nonmissing entries in **V**. When both the input and factor matrices are required to be nonnegative, the problem is called Nonnegative Matrix Factorization (NMF). NMF has been applied to multiple problem domains. NMF also solves the second problem of missing data. Once an algorithm has computed **W** and **H**, the missing values in **V** can be estimated using the corresponding values in **WH**. This process is also called matrix completion, as it 'completes' **V** by predicting the missing entries.

In the movie recommendation setting, the user takes large set of (user, movie, rating) triples as input, where rating is a numeric value between 1 and 5, inclusive. The data is convered into a matrix **V**, where each row corresponds to a specific user, each column corresponds to a specific movie, and entry (i,j) in **V** corresponds to how user i rated movie j. Users have only seen or rated a small number of movies, meaning the dataset contains a substantial amount missing data. When NMF is applied to **V**, the result is a matrix **W** that maps users to genres and a matrix **H** that maps genres to movies. The matrix product **WH** can be used to predict how a user would have scored a movie. If user i_1 and i_2 have given similar review scores, the i_1-th row in **W** and the i_2-th row in **W** will be similar, and so the matrix completion will receive similar recommendations. NMF was successfully used in the Netflix movie recommendation challenge [6] and on a Flickr image recommendation challenge [7].

Standard NMF algorithms assume that the input data is drawn from the low rank linear model, plus mean-zero independent and identically distributed Gaussian noise. However, this assumption may be violated on real world datasets, which may contain anomalies or hostile users trying to harm the performance of the algorithm. In the movie setting, a `review-bomb' or `nuke-attack' is a phenomenon where a group of adversarial users create fake accounts and ratings in order to artificially harm a movie's review score [8,9,10]. In matrix terms, the **H** matrix will be polluted by the adversarial users, and as a result the matrix completion process will give incorrect predictions for the non-adversarial users.

In this paper, the authors consider two existing NMF algorithms and demonstrate their performance suffers under adversarial noise. They then introduce a new algorithm with improved performance on movie recommendation data suffering from a review bomb. The new algorithm alternates between marking elements as corrupt and learning the factor matrices from the noncorrupt entries. The goal of the algorithm is to find a better linear model (i.e. a better **W** and **H**) for data under adversarial noise, which can be used to produce better recommendations.

1.1. Prior Work

Lee and Seung [12] popularized the NMF problem in the setting where **V** is a dense matrix with no missing values and derive a pair of multiplicative update rules. Subsequent follow-up work has considered different objective functions that are more resilient to noise, see [13,14,15]. However, these works focus on the simpler case where **V** has no missing values.

When V has missing values, there are two popular techniques. The first technique by Mao and Saul [16] ignores the missing values when computing update (see equations below). This approach is referred to as the ignore-missing-data approach. Let **Z** be a matrix with **Z**=1 if **V** is present and **Z**=0 if **V** is missing. The Mao and Saul update rules have two components: fix H and update W, then fix **W** and update **H**. These two steps are repeated until convergence.

$$W = W\frac{(Z \circ V)H^T}{(Z \circ (WH))H^T} \qquad H = H\frac{W^T(Z \circ V)}{W^T(Z \circ (WH))}$$

where ○ denotes elementwise multiplication, and division is done elementwise. The Mao and Saul algorithm is theoretically guaranteed to be monotonic in the squared loss, where **Ø** is used to denote missing values, i.e. the summation is over all non-missing values in **V**. Additionally, the Mao and Saul algorithm can be implemented on a distributed Map-Reduce framework [17]

$$\text{Squared loss} = \sum_{V \neq \emptyset} (V - WH)^2$$

The second approach by Zhang et al [18] replaces the missing values with their current values (see below). This approach is referred to as the *replace-missing-data* approach. The Zhang et al approach has three components: fix **H** and update **W**, then fix **W** and update **H**, then replace all missing values with the corresponding value in **WH**. Let **V** be the filled-in matrix with no missing values. Then the Zhang et al update rules are

$$W = W\frac{\overline{V}H^T}{WHH^T} \qquad H = H\frac{W^T\overline{V}}{W^TWH} \qquad \overline{V} = WH \quad \text{for} \quad V = \emptyset$$

When the data is very large, calculating the dense matrix **V** is not practical, and the ignore-missing-data approach is preferable. But for smaller datasets, the replace-missing-data approach has been experimentally shown to offer better performance than the ignore-missing-data approach on data without adversarial noise.

There are many different types of adversarial attacks on machine learning models – see [8-11] for a comprehensive overview of adversarial attacks on non-NMF based recommendation systems. This work focuses on *profile injection* attacks where malicious users can insert bad entries into the dataset. In the movie recommendation setting, this corresponds to adding entries of the form (baduser, movie, score) into the training data, where baduser is an adversarial user. When the raw data is converted into a matrix **V**, the entries in **V** corresponding to these tuples will be called *corrupt entries*. The adversary is not allowed to modify the ratings given by other non-adversarial users.

2. Factorization with Adversarial Noise

To develop a new NMF algorithm to factor data under adversarial noise, a simple rule to identify which elements are created by adversarial users is needed. The input data has already been converted from raw tuples into a matrix **V** with each row corresponding to a user and each column corresponding to a movie.

Let λ > 0 be a user fixed hyper-parameter. We discuss a probabilistic interpretation of λ in Section 2.3. For any given W and H, an element (i,j) is marked as corrupt if the squared error between **V** and **WH** is greater than λ. Intuitively, this choice allows the algorithm to mark an element as corrupt by `paying' a cost of λ. Similar to how missing elements could be ignored or replaced, corrupt elements can either be ignored or deleted. Both approaches are considered.

Algorithm 1: NMF under adversarial noise

```
Convert raw input into a matrix V with missing data
Randomly initialize W and H.
For t = 1, 2, 3, 4, ....
    1) $W = W\frac{(Z \circ \overline{V})H^T}{(Z \circ (WH))H^T}$   $H = H\frac{W^T(Z \circ \overline{V})}{W^T(Z \circ (WH))}$
    2) Mark an entry as corrupt if $(V_{ij} - WH_{ij})^2 > \lambda$
    3) For each corrupt entry, either set Z=0 (to ignore-corrupt-data) or
       set $V = 0.99^t * V + (1-0.99^t) * WH$ (to replace-corrupt-data)
    4) For each missing entry, either set Z=0 (to ignore-missing-data) or
       set V = WH, Z=1 (to replace-missing-data)
```

2.1. Ignore-Corrupt-Data Approach

Just as Mao and Saul's algorithm ignores the missing datapoints when computing **W** and **H**, our algorithm can choose to ignore the corrupted elements when updating **W** and **H**. This approach corresponds to minimizing a clipped loss function. Define ERR(t) to be the standard squared loss and define L(t) to be the clipped squared loss at the very end of iteration t (after step 4 has been completed). The clipped loss (also called the trimmed loss in some settings) has been well established as more robust to outliers and anomalies than the standard squared loss [19,20,21]. The clipped loss has been successfully applied to many machine learning problems, including regression and L(t) replaces errors with value greater than λ with a value of λ, thereby 'clipping' large errors. The clipped error has been successfully applied to other

$$L(t) = \sum_{V \neq \emptyset} max\left[(V - WH)^2, \lambda\right] = \sum_{V \neq \emptyset} max\left[ERR(t), \lambda\right]$$

We now argue L(t) > L(t+1). Let **Z(t)** be the value of **Z** at the end of iteration t: either **Z(t)** = 0 or **Z(t)** =1. Then L(t) can be decomposed into two parts

$$L(t) = \sum_{\substack{V \neq \emptyset \\ Z(t)=1}} ERR(t) + \sum_{\substack{V \neq \emptyset \\ Z(t)=0}} \lambda$$

The Mao and Saul update rules are monotonic, so we have ERR(t) > ERR(t+1). This means that after the **W** and **H** update steps are applied (after step-1 is completed), we have

$$L(t) \geq \sum_{\substack{V \neq \emptyset \\ Z(t)=1}} ERR(t+1) + \sum_{\substack{V \neq \emptyset \\ Z(t)=0}} \lambda = Intermediate(t)$$

We then consider the Z update step. If **Z(t)** = 0 but ERR(t+1) > λ, then setting **Z(t+1)** = 1 replaces a value in Intermediate(t) that was greater than λ with the value λ. Similarly, if **Z(t)** = 1 but ERR(t+1) < λ, then setting **Z(t+1)** = 0 replaces a value of λ in Intermediate(t) with a value that is less than λ. Therefore, we have that

$$Intermediate(t) \geq \sum_{\substack{V \neq \emptyset \\ Z(t+1)=1}} ERR(t+1) + \sum_{\substack{V \neq \emptyset \\ Z(t+1)=0}} \lambda = L(t+1)$$

Combining the above three equations proves the desired result, that L(t) > L(t+1). This proves that the algorithm monotonically decreases the clipped loss.

2.2. Replace-Corrupt-Data Approach

Just as Zhang et al algorithm ignores the replaces datapoints when computing **W** and **H**, our algorithm can choose to replace the corrupt elements when updating **W** and **H**. The seemingly obvious choice would be to replace corrupt elements with the current estimate **WH**. However, during iterations of the algorithm, **WH** is a bad estimation of **V**. Instead, we replace corrupt elements with an interpolation between **V** and **WH**, defined as

V = 0.99 t∗V +(1− 0.99t)∗**WH** for corrupted entries in V

In early iterations, **V** and **V** are very close, and we apply a minimal correction. In later iterations, the algorithm is allowed to make a more substantial correction.

2.3. Probabilistic Interpretation of λ

We can treat λ as an arbitrary user-specified parameter that denotes the `cost' of marking an element as corrupt. However, λ also has a probabilistic interpretation, where the magnitude of λ is defined by the probability an element has undergone adversarial corruption. Assume that the non-corrupt entries of **V** are generated from the underlying low-rank model $\mathbf{V}_{\mathbf{true}}$, plus Gaussian noise with mean 0 and standard deviation σ. Let f be the probability density function of this Gaussian distribution applied to a specific entry (i,j). Then

$$\log f = \log \mathbb{P}\left[V_{true} = v \,|\, V = v, W, H\right] = \log \mathbb{P}\left[Normal(v, \sigma) = WH\right] = a(V - WH)^2 + b$$

where a and b are defined below. We next assume that the probability that entry (i,j) is corrupted is given by p, where $0 < p < 1$. Finally, define

$$\lambda = -\frac{\log p + b}{a} \quad \text{where } a = -\frac{1}{2\sigma^2}, b = -\frac{1}{2}\log(2\pi\sigma)$$

Note that a, b, and log p are all negative, so λ must be positive. In other words, λ is defined by an assumption about the probability an element is corrupt, plus an assumption about the underlying distribution of the non-corrupt data. Additionally, if ERR(t) > λ, then log f < log p and f < p. In other words, one way to interpret Algorithm 1 is that an element is marked as corrupt if the probability that the good distribution generated that element (defined by f defined above) is less than the probability it was corrupted (defined by p). Note that we could have made other assumptions about the good or bad elements, which would have derived a different definition for λ and a different rule for which elements should be marked as corrupt.

3. EXPERIMENTS

To demonstrate the effectiveness of our Corrective NMF algorithm, we run a series of experiments using the same experimental setup used in [8,10,17], We begin with the 1M MovieLens dataset [22] a public dataset of user-movie scores where each user has rated at least twenty movies between one star and five stars. All experiments are run 20 times, and we report averages. All algorithms are implemented in Python using the NumPy library, are run on a 12-core Intel i7 Unix desktop, are given the same random initializations, and are given same amount of time to run. Following the setup of [17], we set k=20, as there are 20 movie genres in the MovieLens dataset.

3.1. Non-Adversarial Noise

We begin by considering two non-adversarial noise models.

- Model 1 – No Noise – We use the MovieLens dataset, unaltered.
- Model 2 – Random Noise – Every entry in the test set flips a coin. With probability 0.1, a value of 1-3 is flipped to a 5, and a value of 4-5 is flipped to a 1.

Recall that the goal of the algorithm is to first estimate a low-rank model from non-missing data, then to use that model to estimate the missing values. To evaluate the performance of the algorithms on non-adversarial noise, we divide the MovieLens scores into a training set (90% of the data), and a test set (10% of the data). In Model 1, the training set is used as-is. In Model 2, the training set undergoes random noise. The training set is then fed as input to each factorization algorithm, which produces the factor matrices **W** and **H**. The elements in the test set are then compared to the predicted values in **WH** to test whether the low-rank approximation is a good model. The evaluation matrix is the normalized mean absolute error defined below, which is the same metric used by Zhang et al in [5].

$$\text{mean abs error} = \frac{1}{5} * \frac{1}{|\text{size of test set}|} * \sum_{\text{test set}} |V - WH|$$

We report the normalized mean absolute error in Table 1. On non-adversarial noise, the ignore-corrupt algorithm has equivalent performance as other algorithms, and the replace-corrupt approach offers a 5-10% improvement over existing algorithms

Table 1: Absolute Error on non-Adversarial MovieLens Data

Algorithm	Model 1 (no noise)	Model 2 (random noise)
Ignore-missing [16]	0.160	0.168
Replace-missing [18]	0.156	0.163
Ignore-corrupt	0.156	0.164
Replace-corrupt	**0.149**	**0.153**
Replace-corrupt, no interpolation	0.155	0.155

3.2. Adversarial Noise

We next consider two adversarial noise models. These models choose a specific ‘target movie’ and will add adversarial noise in an attempt to harm that movie’s rating. Every user in the clean MovieLens dataset has assigned scores to at least twenty movie scores, so the hostile users must also have assigned scores to at least twenty movies, or it would be trivial to detect and delete these adversarial users. The low-knowledge attack simply adds random users to the dataset, while the informed attack adds users who appear similar to legitimate users.

- Model 3 – Low-Knowledge Attack: We augment the dataset with synthetic users. Each synthetic user assigns the target movie a score of 1 and gives nineteen other movies in the dataset a random score between 1-5.
- Model 4 – Informed Attack: We randomly choose existing users and convert them into adversarial users. Adversarial users assign the target movie a score of 1 and leaves their other ratings unaltered.

Recall the assumption that the data has a low-rank model. Define **WH*** to be the factorization that the algorithm would have found if there were no adversarial users. Define **WH** to be the factorization the algorithm finds with adversarial users. To measure the difference between the two models, the *mean absolute prediction shift* is defined as the absolute error between **WH*** and **WH** on the column that corresponds to the targeted movie. This evaluation metric was used in [9, 11] and measures how effective the adversary was at harming the targeting movie’s rating.

Table 2: Mean Absolute Prediction Shift on Adversarial MovieLens Data

Algorithm	Model 3 (low-knowledge)	Model 4 (informed)
Ignore-missing [16]	0.062	0.083
Replace-missing [18]	0.042	0.050
Ignore-corrupt	**0.027**	**0.019**
Replace-corrupt	0.621	0.159
Replace-corrupt, no interpolation	0.467	0.134

On data that has undergone adversarial noise, the ignore-corrupt approach offers a dramatic 64% improvement over the existing algorithms. The replace-corrupt approach does not succeed. Inspecting the data reveals that the replace-corrupt technique fills the missing entries of adversarial users with bad data. In the low-knowledge attack, the adversarial users are imputing random numbers, so it is impossible to estimate what their true values 'should' be in **V**.

4. CONCLUSIONS

In this work, we study the effectiveness of nonnegative matrix factorization under non-Gaussian noise and adversarial noise. Existing NMF algorithms which assume Gaussian noise struggle in the presence of this noise. The new algorithm offers extends existing NMF approaches to find better factorizations under adversarial noise. The key innovation is to define a parameter λ, mark elements with error greater than λ as corrupt, and to either ignore or replace those entries when computing the factorization. Our algorithm is monotonic in a clipped loss function and experimentally outperforms existing NMF techniques.

In this work, we focused on a relatively simple adversarial attacks and a relatively simple rule for marking elements as corrupt. One potential future direction is to consider more sophisticated adversaries and more complex rules for marking elements. Alternatively, while we focus on the standard NMF formulation, the ideas could potentially be applied to other dimensionality reduction techniques like semi-NMF or Bayesian factorization.

REFERENCES

[1] Indyk, Pitor & Motwani, Rajeev (1998) "Approximate Nearest Neighbors: Towards Remoing the Curse of Dimensionality" Proceedings of the thirtieth annual ACM symposium on theory of computing, pp604-613

[2] Oseledts, Ivan & Tyrtyshinikov, Eugene (2009), "Breaking the curse of dimensionality, or how to use SVD in many dimensions", SIAM Journal of Scientific Computing, pp3744-3759

[3] Dempster, Arthur & Laird, Nan & Rubin, Donald (1997) "Maximum likelihood from incomplete data via the EM algorithm", Journal of the Royal Statistical Society, pp1-22

[4] Srebro, Nathan and Jaakkola, Tommi (2003), "Weighted low-rank approximations",Proceedings of the 20th International Conference on Machine Learning, pp720-727

[5] Candes, Emmanuel & Recht, Benjamin (2009), "Exact matrix completion via convex optimization", Foundations of Computational mathematics, pp717

[6] Koren, Yehuda & Bell, Robert & Volinsky, Robert (2009) "Matrix factorization techniques for recommender systems" Computer, pp30-37

[7] Zheng, Nan & Li, Qiudan & Liao, Shengcai & Zhang, Leiming (2010) "Which photo groups should I choose? A comparative study of recommendation algorithms in Flickr", Journal of Information Science, pp733-750

[8] Burke, Robin & O'Mahony, Michael & Hurley, Neil (2015) "Robust Collaborative Filtering" Recommender systems handbook, pp961-995

[9] O'Mahony, Michael & Hurley, Neil & Kushmerick, Nicolas & Silvestre, Guenole (2004), "Collaborative recommendation: A robustness analysis", ACM Transactions on Internet Technology, pp344-377

[10] Sandvig, Jeff & Mobasher, Bamshad & Burke, Robin (2008), "A survey of collaborative recommendation and the robustness of model-based algorithms", IEEE Computer Society Technical Committee on Data Engineering

[11] Mobasher, Bamshad & Burke, Robin & Sandvig, Jeff (2006), "Model-based collaborative filtering as a defense against profile injection attacks", AI Magazine pp1388

[12] Lee, Daniel & Seung, Sebastian, (2001) "Algorithms for nonnegative matrix factorization", Advances in Neural Information Processing Systems, pp556-562

[13] Sra, Suvrit & Dhillon, Inderjit (2006) "Generalized nonnegative matrix approximations with Bregman divergences", Advances in neural information processing systems, pp283-290

[14] Fevotte, Cedric & Idier, Jerome (2011), "Algorithms for nonnegative matrix factorization with beta-divergence, Neural computation, pp2421-2456

[15] Taslaman L & Nilsson B. (2012) "A framework for regularized non-negative matrix factorization, with application to the analysis of gene expression data" PLoS One

[16] Mao, Yun & Saul, Lawrence (2009) "Modeling distances in large scale networks by matrix factorization" ACM SIGCOMM conference in internet measurement, pp278-287

[17] Liu, Chao & Yang, Hung-chih & Fan, Jinliang & He, Li-Wei & Wang, Yi-Min (2010), "Distributed nonnegative matrix factorization for web-scale dyadic data analysis on mapreduce", Proceedings of the 19th international conference on World wide web, pp681-690

[18] Zhang, Sheng & Wang, Weihong & Ford, James & Makedon, Fillia (2006) "Learning from incomplete ratings on nonnegative matrix factorization" SIAM conference on data mining, pp549-553

[19] Yang, Min & Xu, Linli & White, Martha & Schuurmans, Dale, & Yu, Yao-liang (2010) "Relaxed clipping: A global training method for robust regression and classification", Advances in Neural Processing, pp2532-2540

[20] Honore, Bo E (1992), "Trimmed LAD and least squares estimation of truncated and censored regression models with fixed effects", Econometrica: Journal of the Econometric Society, pp533-565

[21] Garcia-Escudero, Luis Angel & Gordaliza, Alfonso (1999), "Robustness properties of k means and trimmed k means", Journal of the American Statistical Association, pp956-969

[22] Harper, F. Maxwell & Konstan, Joseph (2015) "The MovieLens datasets: history and context" ACM Transactions on Interactive Intelligent Systems

AUTHORS

Peter Ballen is a PhD student at the University of Pennsylvania, where he studies matrix factorization algorithms, their theoretical properties, and their applications in data mining.