Challenges in Recommender Systems: Scalability, Privacy, and Structured Recommendations

by

Yu Xin

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2015

© Massachusetts Institute of Technology 2015. All rights reserved.
Challenges in Recommender Systems: Scalability, Privacy, and Structured Recommendations

by

Yu Xin

Submitted to the Department of Electrical Engineering and Computer Science on Mar 16, 2015, in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Computer Science and Engineering

Abstract

In this thesis, we tackle three challenges in recommender systems (RS): scalability, privacy and structured recommendations.

We first develop a scalable primal dual algorithm for matrix completion based on trace norm regularization. The regularization problem is solved via a constraint generation method that explicitly maintains a sparse dual and the corresponding low rank primal solution. We provide a new dual block coordinate descent algorithm for solving the dual problem with a few spectral constraints. Empirical results illustrate the effectiveness of our method in comparison to recently proposed alternatives. In addition, we extend the method to non-negative matrix factorization (NMF) and dictionary learning for sparse coding.

Privacy is another important issue in RS. Indeed, there is an inherent trade-off between accuracy of recommendations and the extent to which users are willing to release information about their preferences. We explore a two-tiered notion of privacy where there is a small set of public users who are willing to share their preferences openly, and a large set of private users who require privacy guarantees. We show theoretically, and demonstrate empirically, that a moderate number of public users with no access to private user information already suffices for reasonable accuracy. Moreover, we introduce a new privacy concept for gleaning relational information from private users while maintaining a first order deniability. We demonstrate gains from controlled access to private user preferences.

We further extend matrix completion to high-order tensors. We illustrate the problem of recommending a set of items to users as a tensor completion problem. We develop methods for directly controlling tensor factorizations in terms of the degree of nonlinearity (the number of non-uniform modes in rank-1 components) as well as the overall number of rank-1 components.
Finally, we develop a tensor factorization for dependency parsing. Instead of manually selecting features, we use tensors to map high-dimensional sparse features into low dimensional (dense) features. Our parser achieves state of the art results across multiple languages.

Thesis Supervisor: Tommi Jaakkola
Title: Professor
Acknowledgments

First and foremost, I would like to thank my advisor Prof. Tommi Jaakkola for his continuous support throughout my graduate years. He has been a great advisor and one of the smartest people I know. Whenever I encountered a problem in my research, Tommi would show me not just one, but a few different ways to solve the problem. I hope to one day gain the insight he has in this field.

I would also like to thank my committee members Alan Willsky and Leslie Kaelbling for their guidance on my thesis.

Next, I would like to thank the many smart and interesting people I met at MIT. I benefited a lot from discussions with my collaborators Tao Lei, Yuan Zhang and Yuan Luo. I also had a lot of fun with my officemates Andreea Gane, Paresh Malalur and David Reshef who made the office enjoyable. I am very grateful to Wujie Huang for his friendship over 10 years. The two of us were undergrads at Tsinghua, are currently attending MIT and will be joining the same company afterwards. I hope we will always be friends.

Finally, my biggest thanks go to my parents Xiaozhu Wang and Ziliang Xin for their infinite support, continuous encouragement and unconditional love they have always given me.
# Contents

1 Introduction .................................................. 15

1.1 Challenges .................................................. 17

1.1.1 Scalability .................................................. 18

1.1.2 Privacy ..................................................... 19

1.1.3 Structured recommendations ............................. 20

1.2 Summary of contributions .................................. 21

2 Background ...................................................... 23

2.1 Matrix Factorization .......................................... 24

2.2 Optimization algorithms ...................................... 26

2.2.1 Proximal gradient ......................................... 27

2.2.2 Semi-definite programming ............................... 28

2.3 (Differential) privacy ......................................... 29

2.4 Tensor factorization .......................................... 32

3 Primal dual algorithm ........................................ 35
3.1 Derivation of the dual ........................................... 37
3.2 Solving the dual .................................................. 40
3.3 Constraints and convergence .................................. 43
3.4 Non-negative matrix factorization .............................. 44
3.5 Sparse matrix factorization ...................................... 48
3.6 Independent Component Analysis ............................... 52
3.7 Experiments ........................................................ 54
  3.7.1 Primal-dual algorithm for matrix completion ............ 54
  3.7.2 Non-negative matrix factorization ........................... 57
  3.7.3 Sparse matrix factorization .................................. 60

4 Privacy mechanism .................................................. 63
  4.1 Problem formulation and summary of results ................. 64
  4.2 Analysis .......................................................... 68
    4.2.1 Statistical Consistency of \( \hat{\Sigma} \) ...................... 68
    4.2.2 Prediction accuracy ......................................... 70
  4.3 Controlled privacy for private users .......................... 71
  4.4 Experiments ...................................................... 75
  4.5 Conclusion ....................................................... 79

5 Tensor factorization for set evaluation ......................... 81
  5.1 Notation .......................................................... 82
# List of Figures

1-1 Examples of Recommendation Systems ........................................ 17
2-1 Illustration of a statistical database ........................................ 29
2-2 Illustration of a statistical database ........................................ 31
2-3 Example of a 3-dimensional tensor ........................................ 32
2-4 CP decomposition (Kolda and Bader, 2009) .............................. 33
3-1 test RMSE comparison of JS and PD ........................................ 56
3-2 Optimization progress of JS and PD ........................................ 56
3-3 test RMSE comparison of GECO and PD ................................. 57
3-4 Log-likelihood comparison of NMF and Bigram for different document
    groups ..................................................................................... 60
3-5 Left: Log-likelihood of NMF as a function of time; Right: Time as a
    function of rank. ..................................................................... 60
3-6 test RMSE comparison of Trace norm, Trace norm w. $L_0$, Trace norm
    w. $L_1$ regularization .................................................................. 61
4-1 Distributed system setup ......................................................... 65
4-2 Summary of operations on both sides ........................................... 67
4-3 Example of $\mathcal{M}(r)$ ................................................................. 74
4-4 Test RMSE as a function of users and ratings ................................. 76
4-5 Test RMSE as a function of private user numbers. **PMC**: the privacy mechanism for continuous values; **PMD**: the privacy mechanism for discrete values; **Lap eps=1**: DP with Laplace noise, $\epsilon = 1$; **Lap eps=5**: same as before except $\epsilon = 5$; **SSLP eps=5**: sampling strategy described in (Duchi et al., 2013) with DP parameter $\epsilon = 5$; **Exact 2nd order**: with exact second order statistics from private users (not a valid privacy mechanism); **Full EM**: EM without any privacy protection. . 77
5-1 Test RMSE compared to linear regression, nonlinear regression model with second order monomials and PARAFAC tensor factorization method. The tensor completion approach with $d=2$ consistently performs better than the three methods with improvement ($10^{-2}$): $2.66 \pm 0.26$, $1.77 \pm 0.33$ and $3.55 \pm 0.86$ ......................................................... 99
5-2 Test RMSE as a function of regularization for various $d$ ............... 99
6-1 Average UAS on CoNLL testsets after different epochs. Our full model consistently performs better than NT-1st (its variation without tensor component) under different choices of the hyper-parameter $\gamma$. .............. 109
List of Tables

1.1 Example of collaborative filtering ........................................... 18

6.1 Word feature templates used by our model. pos, form, lemma and morph stand for the fine POS tag, word form, word lemma and the morphology feature (provided in CoNLL format file) of the current word. There is a bias term that is always active for any word. The suffixes -p and -n refer to the left and right of the current word respectively. For example, pos-p means the POS tag to the left of the current word in the sentence. .................................................. 103

6.2 First-order parsing (left) and high-order parsing (right) results on CoNLL-2006 datasets and the English dataset of CoNLL-2008. For our model, the experiments are ran with rank $r = 50$ and hyper-parameter $\gamma = 0.3$. To remove the tensor in our model, we ran experiments with $\gamma = 1$, corresponding to columns NT-1st and NT-3rd. The last column shows results of most accurate parsers among (Nivre et al., 2006; McDonald et al., 2006; Martins et al., 2010, 2011, 2013; Koo et al., 2010; Rush and Petrov, 2012; Zhang and McDonald, 2012; Zhang et al., 2013) . . 106

6.3 Results of adding unsupervised word vectors to the tensor. Adding this information yields consistent improvement for all languages. . . 109
6.4 The first three columns show parsing results when models are trained without POS tags. The last column gives the upper-bound, i.e. the performance of a parser trained with **12 Core POS tags**. The low-rank model outperforms NT-1st by a large margin. Adding word vector features further improves performance.

6.5 Five closest neighbors of the queried words (shown in bold). The upper part shows our learned embeddings group words with similar syntactic behavior. The two bottom parts of the table demonstrate how the projections change depending on the syntactic context of the word.

6.6 Comparison of training times across three typical datasets. The second column is the number of tokens in each data set. The third column shows the average sentence length. Both first-order models are implemented in Java and run as a single process.
Chapter 1

Introduction

Personalization has become one of the key features of on-line content. For instance, Amazon frequently recommends items to customers based on their previous purchase history. Similarly, Facebook ranks and displays news feeds as to match personal interests of their users. The engines behind personalization are known as recommendation systems (RS). These systems analyze the behavioral patterns of users in an attempt to infer user preferences over artifacts of interest. Figure 1-1 gives two examples of recommender systems (RS).

Most algorithms underlying RS can be seen as focusing on either content-based filtering (CB), collaborative filtering (CF), or some combination of the two. CB assumes that each item to be recommended has an associated feature vector that can be used for prediction either directly or by means of evaluating similar other items. To be successful, similarity based methods require additional control of diversity to avoid overwhelming users with many items that they are already familiar with (Adomavicius and Tuzhilin, 2005).

CF in its pure form treats both users and items as symbols without assuming any additional feature representations. Nevertheless, CF methods can be used to complete the limited information about a particular user by borrowing purchase/rating histories
of other similar users. In this sense recommendations are based on like-minded other users rather than direct similarities between items. As a preprocessing step, user histories are first transformed into numerical values or ratings that represent users’ evaluations of items. For instance, the ratings can be the number of stars users give to movies at IMDB. They can also be binary values indicating if users select “like” or “dislike” for a Youtube video. In general, the task is to make accurate predictions of unknown ratings in the user-item table. Table 1.1 gives a simple example of movie recommendation with user provided ratings on the scale of zero to five. The question mark represents an unknown rating. In the example, users A and B have similar past ratings, giving us the means to recommend movie “Interstellar” to A by borrowing the rating from user B.

Various criteria have been proposed for evaluating RS performance. Traditional measures such as mean absolute error (MAE) or root mean squared error (RMSE) are among the most commonly used, primarily due to their simplicity. Ranking measures that evaluate the order of recommended items better match the actual task since the predicted items are displayed as ordered lists (Herlocker et al., 2004; Balakrishnan and Chopra, 2012). Besides the accuracy of predicting ratings, or their rank, another important factor is diversity (Zhou et al., 2010). Methods that only highlight popular, easily found items are not useful. The real value of RS comes from finding items that users would like yet be difficult to find otherwise. While important, we will not consider diversity in this thesis.

The main statistical difficulty in RS comes from the fact that observations are sparse in the sense that each user has rated only a small portion of the items. In the extreme case, a newly joined user has no previous ratings. To provide useful predictions even for new users, Melville et al. (2002) propose a hybrid method of CF and CB where items have additional attributes (beyond their ids) that can be relied upon. Ma et al. (2011) use data from social network to impose additional regularization on CF. Our focus in this thesis on CF alone, without access to additional features.
Most basic CF algorithms assume that users' interests are static and therefore increasingly recoverably with additional observation. Of course, in practice, user interests change with time (Koychev and Schwab, 2000). Classical time-window approaches do not work well to offset this problem since the data are already sparse. Koren (2010) propose a method that separates transient effects from long term patterns, i.e., explicitly modeling time dependent behavior of users. As we focus on the optimization and theoretical results, we will adopt the static user assumption in this thesis.

Figure 1-1: Examples of Recommendation Systems

1.1 Challenges

While RS have been widely adopted and studied for over a decade, a few key challenges remain. The focus of this thesis is on three of these challenges: scalability, privacy,
Table 1.1: Example of collaborative filtering

<table>
<thead>
<tr>
<th>User</th>
<th>Gone Girl</th>
<th>Interstellar</th>
<th>The Hobbit</th>
<th>Big Hero</th>
<th>X-Men</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>4</td>
<td>?</td>
<td>?</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>B</td>
<td>4</td>
<td>5</td>
<td>?</td>
<td>5</td>
<td>?</td>
</tr>
<tr>
<td>C</td>
<td>?</td>
<td>5</td>
<td>4</td>
<td>?</td>
<td>5</td>
</tr>
</tbody>
</table>

1.1.1 Scalability

The amount of data used as input to RS is growing quickly as more users and items are added. For a popular website, for example, the size of stored user behavior data can easily reach TBs per day. Despite the large amount of data, most RS aspire to respond interactively in less than a second in order to keep users engaged. A key challenge here is to design efficient learning algorithms that can handle such large scale datasets.

CF algorithms cannot operate on users independently. Instead, the algorithms have to learn jointly from the experience of all users; this is time consuming with large numbers of users. For example, $k-NN$ method (Resnick et al., 1994) first finds a neighborhood for each user by thresholding similarity scores or finding $k$ most similar users. It then generates predictions by calculating weighted average of the neighboring users’ ratings. Because similarities among all pairs of users are computed, the running time of the method is quadratic in the number of users.

Several approaches have been proposed to deal with the scalability issue. Das et al. (2007) use an online learning algorithm that processes each user and updates parameters sequentially. The online learning algorithm is generally more efficient than the batch method as it incorporates updates immediately instead of waiting to process the cumulative batch of data from all users at once. Gemulla et al. (2011) propose a distributed algorithm in which most of the computations could be conducted on multiple machines in parallel.
Most RS can be modeled as a matrix completion problem with rows being users, columns being items and entries being ratings (Koren et al., 2009; Srebro et al., 2003). The goal is to complete the matrix given partially observed entries. In this thesis, we focus on a particular method for matrix completion that uses trace norm regularization (Jaggi and Sulovsk, 2010; Xin and Jaakkola, 2012; Yang and Yuan, 2013; Ji and Ye, 2009). The model is desired because the optimization problem associated with it is convex. The convexity guarantees that any local minimum is also a global minimum, and therefore the final result does not depend on the initialization and its convergence property can be theoretically analyzed. However, the associated convexity creates a difficulty in optimization. Many proposed algorithms only work for small data sets. We propose a new primal dual algorithm that improves the efficiency for large data sets significantly.

### 1.1.2 Privacy

With an understanding of the value of user data, most websites are collecting as much user data as possible. This approach raises privacy concerns because the data may contain sensitive information that the users wish to keep private, e.g. users’ addresses and payment history. Although occasionally users are presented with privacy policies concerning the usage of data, they usually have no explicit control over the data.

Most research on privacy protection focuses on the task of publishing data from a central database such as medical records collected by hospitals. The privacy mechanisms can be separated into two types: interactive and non-interactive. In the non-interactive setting, a “sanitized” version of the data are published. All following operations will be conducted on the sanitized version. Methods in this setting involve data permutation and removing identifier information such as names and addresses (Sweeney, 2002; Machanavajjhala et al., 2007). However, without specified utilities of the data, general sanitization methods may not obtain satisfying results (Chawla et al., 2005). In addition, the sanitized data could still be used to identify users when
side information is provided. For example, Netflix released user rating data for their recommendation competition by removing all identifiers. Barbaro et al. (2006) show that the Netflix users can still be identified by referencing public IMDB user data due to the fact that it is rare for two different users to share a very similar taste in movies.

In contrast, interactive setting allows users to pose queries about the data and receive (possibly noisy) data. A commonly used concept in this setting is Differential Privacy (DP). DP guarantees that in the query response, it is difficult to determine if a user is in the database or not. Unlike the anonymity approach, privacy is guaranteed even if an adversarial user has additional background information. The most popular method of implementing DP is by adding noise to the query response. The level of noise depends on the sensitivity of the query function to a single data record. For statistical queries such as finding the mean or maximum, it has been proven that simply adding Laplace noise will suffice.

Despite the nice theoretical results, setting up a secure central database for RS that holds all user data is difficult. In particular, it requires users to release their own data to a trusted system that operates in a restricted manner. Considering the potential business interests that are involved and the complexity of restrictions applied to the system, the setting can be unrealistic. In this thesis, we consider a distributed setting. Each user protects his data in a personal computer. They only share limited information to a central server. Thus privacy can be preserved. We will specify the privacy notation and mechanisms in chapter 4.

1.1.3 Structured recommendations

Current RS predict individual items that users may want. An interesting extension is to predict preferences for sets of items. For instance, if the system figures out that a user is going skiing for the first time, it can recommend a pair of boots, goggles, helmet and ski pants that have matching colors and price levels. In this way, a user
can obtain everything that he or she needs for skiing with a single purchase.

There are two challenges in structured recommendations. First, the number of possible sets grows exponentially with the group size. Considering that the number of items is already very large, the efficiency of learning algorithms could be an issue. Second, unlike individual items, it is unclear how to select the right score function for sets. For instance, a simple approach for approximating the scoring of a set is to find the average of the score of items in the set. Then the problem reduces to learning a score for each individual item and thus can be solved efficiently. However, this approach is weak because it ignores interactions between items.

To incorporate higher order interactions in score function for sets, one can use a tensor regularization framework. Tensors are generalizations of vectors and can be represented as multi-dimensional arrays. By definition, a first order tensor is a vector, a second order tensor is a matrix, and so on. Each axis in a tensor is called a mode. Common matrix concepts such as rank and maximum eigenvalue are also defined for tensors. Tensors are more powerful than matrices because they permit one to model high order interactions among its modes. However unlike matrices, computing these values are usually NP-hard (Hillar and heng Lim, 2009). In chapter 5, we propose a new convex regularizer for tensor that allows us to explicitly control the order of interactions among items in sets.

### 1.2 Summary of contributions

We summarize here the main contributions of this thesis addressing many of the key challenges raised in the previous section.

1. **We propose an efficient primal dual algorithm for CF with trace norm regularization.**

The regularization problem is solved by utilizing a constraint generation method that
explicitly maintains a sparse dual and the corresponding low rank primal solution. We provide a new block coordinate descent algorithm for solving the dual problem with few spectral constraints. Empirical results illustrate the effectiveness of our method in comparison to recently proposed alternatives. We also extend the method to non-negative matrix factorization, sparse coding and independent component analysis.

2. We develop a new privacy mechanism for RS

We explore a two-tiered notion of privacy where there is a small set of “public” users who are willing to share their preferences openly, and a large set of “private” users who require guaranteed privacy. We show theoretically and demonstrate empirically that a moderate number of public users with no access to private user information suffices for reasonable accuracy. Moreover, we introduce a new privacy concept for gleaning additional information from private users while maintaining a first order deniability. We demonstrate gains from controlled access to private user data.

3. We build a tensor model for set recommendations

We formulate the scores of sets via tensors. With a new convex regularizer of tensors, we can explicitly control the order of interactions modeled in the score function. We develop an efficient algorithm for estimating the tensor efficiently that iteratively adds low rank-1 tensors. We analyze statistical consistency of the estimation problem and apply the method to set recommendations.

We apply tensors to dependency parsing. By constraining tensors to have low rank, we obtain low dimensional embeddings that are tailored to the syntactic context of words. Our parser consistently outperforms the Turbo and MST parsers across 14 languages. We also obtain the best published unlabeled attachment scores (UAS) results on 5 languages.
Chapter 2

Background

Observations in RS consist of user ID, item ID and rating value. The rating value indicates the user’s preference about the item. The observation data can be represented in a matrix format with rows being users, columns being items, entries being the corresponding rating values. As was discussed in the previous section, each user has only a few ratings, so the matrix is sparsely filled. For instance, in Netflix recommendation challenge dataset, the averaged user rated around 200 movies within 17770 total movies. Mathematically, the goal is to predict and fill in the missing entries based on the observed entries. The recommended items are those with the highest predicted ratings.

In general, prediction accuracy relates to the number of observed ratings. Since in many RS the ratio of observed ratings can be well below 1%, strong regularization is required during estimation. In the following, we discuss in detail most commonly used assumptions and regularizations for RS.
2.1 Matrix Factorization

Consider RS with $n$ users and $m$ items. The underlying complete rating matrix to be recovered is $\hat{X} \in \mathbb{R}^{n \times m}$. We assume that entries are observed with noise. Specifically,

$$Y_{i,j} = \hat{X}_{i,j} + \epsilon_{i,j}, \quad (i, j) \in \Omega$$

(2.1)

where $\Omega$ is the set of observed ratings and noise is i.i.d and follows a zero-mean sub-Gaussian distribution with parameter $\|\epsilon\|_{\psi_2} = \sigma$. We refer to the noise distribution as $Sub(\sigma^2)$ (see (Vershynin, 2010)). For each observed rating, we have a loss term that is a penalty based on the difference of the observed rating to the predicted rating. To predict missing ratings, additional assumptions are required.

A typical assumption in RS is that only a few underlying factors affect users’ preferences. Mathematically, the assumption suggests that matrix $\hat{X}$ can be divided into a product of two smaller dimensional matrices, $UV^T$, where $U \in \mathbb{R}^{n \times d}$ and $V \in \mathbb{R}^{m \times d}$ with $d \ll m, n$ (Fazel et al., 2001; Srebro et al., 2005). $U$ is the user factor matrix and $U_i$ is a $d$-dimensional feature column vector for user $i$. Correspondingly, $V_j$ is the $d$-dimensional feature vector for item $j$. Their inner product $U_i V_j^T$ is then user $i$’s rating value for item $j$.

It is usually beneficial to add additional regularizations to $U$ and $V$ in order to avoid over-fitting. The optimization problem is then given by,

$$\min_{U \in \mathbb{R}^{n \times d}, V \in \mathbb{R}^{m \times d}} \sum_{(i,j) \in \Omega} \text{Loss}(Y_{i,j}, U_i V_j^T) + \lambda \|U\|^2_F + \lambda \|V\|^2_F$$

(2.2)

The loss function is typically convex and becomes zero when $Y_{i,j} = U_i V_j^T$. The most popular choice is the squared loss, which corresponds to assuming that the noise is Gaussian (Srebro et al., 2003). Other loss functions such as hinge loss and absolute error have also been used (Srebro et al., 2005).

The factorization assumption requires $\hat{X}$ to have rank of at most $d$. This usually
referred to as the low rank assumption. The factorization assumption has a few clear advantages. First, \( U \) and \( V \) are more efficient representation than the full matrix \( \hat{X} \). Second, the associated optimization problem is easier due to a smaller quantity of variables. Finally, computation of the prediction \( U_i^T V_j \) is efficient which only takes \( O(d) \) operations.

The optimization problem in (2.2) can be solved approximately by an alternating minimization algorithm. In each iteration, \( V \) is fixed and minimizing \( U_i \) reduces to a simple linear regression. Then \( U \) is fixed and \( V \) is minimized. The algorithm continues until \( U \) and \( V \) converge. It can be shown that the objective decreases in each iteration and that the algorithm is guaranteed to converge. However, the objective is not convex in \( U \) and \( V \) jointly. Consequently, the algorithm is only guaranteed to find a local optimum while finding a global optimum is hard (Srebro et al., 2003).

The resulting local optimum depends on the initialization. As initialization is typically random, the results vary. Additionally, it is also difficult to theoretically analyze the performance of the algorithm. Non-convexity arises from the low rank constraint. Its convex relaxation is a trace norm (a.k.a nuclear norm) that is a 1-norm penalty on the singular values of the matrix (Srebro et al., 2004b)

\[
\|X\|_* = \sum_i s_i(X)
\]  

(2.3)

where \( s_i(x) \) is the \( i \)th singular value of \( X \). With sufficient regularization, trace norm leads to a low rank solution. A trace norm is also associated with the factorizations since it can be shown that

\[
2\|X\|_* = \min_{U, V} \|U\|_F^2 + \|V\|_F^2, \text{ s.t. } UV^T = X
\]  

(2.4)

Notice that the minimization does not constrain the dimensionality of \( U \) and \( V \), therefore the regularization term in (2.2) is an upper bound on \( \|X\|_* \). Replacing the
regularization term of $U$ and $V$ with a trace norm, we get

$$
\min_{X \in \mathbb{R}^{n \times m}} \sum_{(i,j) \in \Omega} \text{Loss}(Y_{i,j}, X_{i,j}) + \lambda \|X\|_*
$$

(2.5)

The optimization is now convex, thus all local optima are also global optima.

### 2.2 Optimization algorithms

One key difficulty with the trace norm regularization approach is that while the resulting optimization problem is convex, it is not differentiable. A number of approaches have been suggested to deal with this problem (e.g., Beck and Teboulle, 2009). In particular, many variants of proximal gradient methods (e.g., Ji and Ye, 2009) are effective as they are able to fold the non-smooth regularization penalty into a simple proximal update that makes use of a singular value decomposition. An alternative strategy would be to cast the trace-norm itself as a minimization problem over weighted Frobenius norms that are both convex and smooth.

Another key difficulty arises from the sheer size of the full rating matrix, even if the observations are sparse. This is a problem with all convex optimization approaches (e.g., proximal gradient methods) that explicitly maintain the predicted rating matrix (rank constraint would destroy convexity). The scaling problem can be remedied by switching to the dual regularization problem in which dual variables are associated with the few observed entries (Srebro et al., 2005). The standard dual approach would, however, require us to solve an additional reconstruction problem (using complementary slackness) to realize the actual rating matrix. Next, we discuss two algorithms that are particularly designed for this optimization problem.
2.2.1 Proximal gradient

Non-smooth objectives frequently occur in sparse regularization problems such as Lasso and group Lasso (Tibshirani, 1996; Meier et al., 2008). A general method that uses proximal gradient (Parikh and Boyd, 2013), a substitute of gradient, has shown promising results across a variety of problems. The method assumes the objective can be split into two components:

$$
\min_x \ g(x) + h(x) \quad (2.6)
$$

where $g(x)$ is convex and differentiable, and $h(x)$ is convex and possibly non-differentiable. The proximal mapping of $h(x)$ is

$$
\text{prox}_h(x) = \arg\min_u \left( h(u) + \frac{1}{2} \|u - x\|_2^2 \right) \quad (2.7)
$$

Then the proximal gradient algorithm updates $x^k$ in the $k$-th iteration as

$$
x^k = \text{prox}_{t_k h}(x^{k-1} - t_k \nabla g(x^{k-1})) \quad (2.8)
$$

where $t_k > 0$ is the step size determined by line search. It is easier to interpret the algorithm by expanding the proximal operator,

$$
x^k = \arg\min_u \left( h(u) + g(x^{k-1}) + \nabla g(x^{k-1})^T (u - x^{k-1}) + \frac{1}{2t_k} \|u - x^{k-1}\|_2^2 \right) \quad (2.9)
$$

Indeed the algorithm finds $x^k$, and minimizes a quadratic approximation to the original objective around $x^{k-1}$. The analysis of proximal gradient method is similar to that of the traditional gradient method. With appropriate step size $t_k$, the algorithm is guaranteed to converge to the global optimum. A typical choice for $t_k$ is the Lipschitz constant of the gradient of $g(x)$ (Parikh and Boyd, 2013).

The proximal gradient method requires that the proximal mapping can be computed efficiently. Let $X \in \mathbb{R}^{n \times n}$ be a matrix with singular vector decomposition (SVD)
\( X = PSQ^T \) with \( S = \text{diag}(s_1, \ldots, s_n) \), \( P \) and \( Q \) being unitary, we have

\[
\text{prox}_{\ell_2\|X\|}(X) = P\text{diag}(\eta_t(s_1), \ldots, \eta_t(s_n))Q^T \tag{2.10}
\]

where \( \eta_t(s) = \max(s - t, 0) \) is the soft-thresholding operator (Toh and Yun, 2010). The operator removes all the singular values that are less than \( t \), thus generating a low rank solution. Despite the simplicity of the algorithm, the key step in SVD takes \( O(n^3) \) computations. Although a few improvements have been proposed that consider the intermediate solution \( X^k \) has low rank, the algorithm remains slow for large \( n \).

### 2.2.2 Semi-definite programming

Another method is to transform the original problem (2.5) into a semi-definite program (SDP) (Jaggi et al., 2010). Assuming \( X = UV^T \), by introducing \( Z = \begin{pmatrix} U \\ V \end{pmatrix} \begin{pmatrix} U^T & V^T \end{pmatrix} \) such that \( \text{tr}(Z) = \|U\|^2_F + \|V\|^2_F \), a constrained version of (2.5) is

\[
\min_{Z \in S(n+m) \times (n+m), \ Z \succeq 0, \ \text{tr}(Z)=t} \sum_{(i,j) \in \Omega} \text{Loss}(Y_{i,j}, Z_{i,j+n}) \tag{2.11}
\]

Where \( S \) is the set of symmetric matrices and \( Z \succeq 0 \) means \( Z \) is positive semi-definite. The advantage of this transformation is that many SDP algorithms can be applied (Srebro et al., 2004b). In particular, the algorithm proposed in (Hazan, 2008) is suitable for the trace constraint. In each iteration, the gradient of the objective \( G(Z) \) is computed. \( G(Z) \) is a symmetric matrix of the same size as \( Z \) with dominant eigenvector \( v \). Instead of updating \( Z \) along \( G(Z) \), the algorithm updates along \( vv^T \),

\[
Z^k = Z^{k-1} + \alpha(tvv^T - Z^{k-1}) \tag{2.12}
\]

where \( \alpha \) is the step size that can be fixed or optimized. It is easy to show that the update satisfies the trace constraint \( \text{tr}(Z^k) = t \).
Compared to the proximal gradient approach, the major step is finding the maximum eigenvector $v$, which can be solved efficiently by the power method. Therefore the algorithm is suitable for large scale RS. However, the algorithm is designed for SDP and not specifically tailored to trace norm regularization. In the next chapter, we will describe a novel primal dual algorithm that shows significant improvements, both theoretically and empirically.

### 2.3 (Differential) privacy

Besides scalability, privacy in RS has recently become a key research question. The concepts originate from the need to ensure guarantees when releasing data for a centralized database, such that no original records can be revealed. One of the most popular frameworks in the privacy literature is Differential Privacy (DP) (Dwork et al., 2006; Dwork, 2006; Alvim et al., 2012; Duchi et al., 2012).

DP assumes a database that users send statistical queries to and receive answers from as illustrated in figure 2-1. The queries can be, for example, finding the maximum or mean of a particular column in the database. Assuming some of the users are adversarial, the privacy goal in this setting is to prevent adversaries from inferring any of the original records based on answers they receive.

![Figure 2-1: Illustration of a statistical database](image-url)
A precise answer to a query could potentially reveal original records. For instance, assuming there are only two records in the DB, and user already knows one but not the other. By asking the mean of the two records, the user can easily infer the value of the other record. To protect privacy, the answers have to be masked with noise in this setting.

One guarantee that has been proposed to achieve privacy is $k$-anonymity (Sweeney, 2002). Assuming each record in the database contains information about a different person, $k$-anonymity requires that a person cannot be distinguished from another $k-1$ individuals in the database. The indistinguishableness gives deniability to each persona thus protecting privacy to a certain extent. However, the approach does not work for a high dimensional database because an unacceptably high amount of information would be lost (Aggarwal, 2005).

DP offers the privacy guarantee that it is difficult to differentiate if a record is contained in the database or not. Mathematically, let $A$ be the answer to a query, $r_i$ be the $i$-th record, then

$$\exp^{-\epsilon} \leq \frac{\Pr(A = \alpha | r_i \in \text{DB})}{\Pr(A = \alpha | r_i / \in \text{DB})} \leq \exp^\epsilon, \ \forall i, \ \forall \alpha$$

(2.13)

where $\epsilon$ is a small constant that controls the degree of privacy preservation. The difference of the two conditional distributions in (2.13) is illustrated in Figure 2-2. The blue and red lines indicate the distributions of query answers of two databases that differ by one record. Notice that the constraint is required to hold for all records and all $\alpha$.

It turns out that for most queries, DP can be achieved by simply adding Laplace noise to the exact answers (Dwork et al., 2006). Let $\hat{\alpha}$ and $\hat{\alpha}_{-i}$ be the exact answers with and without record $i$, and let $b$ be the sensitivity of the query function defined as the maximum absolute change of the function by varying one record in the database.
Then the noisy answer is

\[ A = \hat{\alpha} + \epsilon, \; \epsilon \sim \text{Lap}(0, \frac{b}{\epsilon}) \]  \hspace{1cm} (2.14)

To see the the noisy answer satisfies DP,

\[ \frac{\Pr(A = \alpha | r_i \in \text{DB})}{\Pr(A = \alpha | r_i \notin \text{DB})} = \frac{\Pr(\epsilon = \alpha - \hat{\alpha})}{\Pr(\epsilon = \alpha - \hat{\alpha} - |i|)} \leq \exp^{\epsilon |\hat{\alpha} - \hat{\alpha} - |i|} \leq \exp^{\epsilon} \]  \hspace{1cm} (2.15)

The other direction can be proved similarly. The result can be easily extended to multiple queries by computing the posterior distribution. For \( k \) queries, the DP parameter is \( k\epsilon \). Therefore, the more amount of queries an adversary can make, the larger amount of noise has to be added.

DP is a very strict requirement. The amount of noise it requires may be so large such that the answer does not contain any useful information, especially if the database is small. In chapter 4, we introduce another privacy mechanism whose guarantees better fits RS.
2.4 Tensor factorization

In RS, access to side information such as time, user demographics or item description often improves prediction accuracy. For instance, season is usually important for recommending clothes. One way to use side information is to build a complementary model solely based on the side information, then combine this model with collaborative filtering. However, interactions of side information with user or item are ignored.

Another approach is to treat side information equally along with user and item ids. In the example of recommending clothes, the data are tuples [user, item, season, rating]. We can arrange such data into a multi-dimensional array, also called a tensor. Figure 2-3 shows an example of a 3-dimensional tensor. Each dimension is referred to as a mode.

One way to construct a tensor is using the Kronecker product, denoted as \( \otimes \), that is an extension of the outer product. For example, a rank one matrix \( T = ab^T \) can also be represented as \( T = a \otimes b \). Similarly, a tensor that is composed of a Kronecker product of only vectors has rank one. Any tensor \( T \) can be decomposed into a sum of rank-1 tensors. Assume \( T \) has 3-modes, then

\[
T = \sum_i a_i \otimes b_i \otimes c_i
\]  

(2.16)
To represent a tensor efficiently, we may approximate it as a sum of at most R rank-1 tensors as illustrated in Figure 2-4. If we use $L_2$ error to measure the approximation error, the approach is called CP-decomposition and is considered an extension of SVD for a matrix (Kolda and Bader, 2009).

![Figure 2-4: CP decomposition (Kolda and Bader, 2009)](image)

Low rank constraint is commonly used as a regularization in optimization problems (Karatzoglou et al., 2010; Rendle and Schmidt-Thieme, 2010b). Although the objective with an explicit rank constraint is non-convex, a local optimum is easy to find. In Chapter 5, we introduce a new regularizer that is a convex relaxation to the rank constraint. We then apply the regularizer to set recommendation problems. In Chapter 6, we use low rank tensor for dependency parsing and obtain the state of the arts results.
Chapter 3

Primal dual algorithm

Matrix completion aims to make predictions on the missing entries of a rating matrix based on its sparsely observed entries. Since only limited information is available about each user, strong regularity assumptions are needed about the underlying rating matrix. For example, we may assume that the rating matrix has low rank, or introduce a trace norm as a convex regularizer (1-norm over the singular values of the matrix) Fazel et al. (2001); Srebro et al. (2004a). A number of algorithms have been developed for solving such, convex, regularization problems (e.g., Srebro et al. (2004a); Ji and Ye (2009); Jaggi and Sulovsk (2010); Xin and Jaakkola (2012)).

The assertion of low rank assumption has been applied beyond regular matrix completion. One such example is Non-negative Matrix Factorization (NMF) which intends to factorize a matrix into a product of two low dimensional non-negative matrices (Lee and Seung, 1999, 2001). It comes from the observation that most data is inherently non-negative and some data also requires additive property of different components. NMF is widely used in face recognition (Lee and Seung, 1999; Hoyer, 2004), document clustering (Xu et al., 2003; Shabnaz et al., 2006), and gene expression data analysis (Gao and Church, 2005; Kim and Park, 2007). Ding et al. (2008) have shown that NMF is closely related to Probabilistic Latent Semantic Indexing (PLSI) with a particular loss function.
The objective function of NMF is non-convex and therefore local optima exist. The optimization problem can be solved by an alternating minimization algorithm that iteratively optimizes each factor, followed by projecting to non-negative space (Kim and Park, 2007). Lee and Seung (2001) propose a different method that involves multiplicative updates that decreases the objective function while preserving non-negativity at the same time. In general, because of non-convexity of the objective function, solving NMF exactly is NP-hard (Vavasis, 2009). A side effect of NMF is that its resulting factors are usually sparse. This is because during optimization, many variables reach the boundary of constraints are automatically set to zeros. Hoyer (2004) shows that by adding an additional minimum limit on entry values, we can explicitly control the sparsity of resulting factors for easy interpretation.

Another variety of matrix factorization is dictionary learning for sparse coding which imposes a sparsity regularization constraint on one of the factors (Lee et al., 2007; Mairal et al., 2010a). Recently, it has been applied to image de-noising (Mairal et al., 2008), image classification (Mairal et al., 2009; Yang et al., 2009) and audio processing (Zibulevsky and Pearlmutter, 2001). The dictionary serves as a set of bases for reconstruction, and the corresponding coding is sparse such that only a few bases are active. The model was originally used to understand neural coding from sensor signals (Olshausen and Field, 1996). It enjoys a few desirable properties including providing high dimensional and nonlinear representations and allowing over-complete bases.

The major optimization difficulty of sparse coding comes from $L_1$ regularization. A number of efficient algorithms have been suggested to solve the problem. Lee et al. (2007) propose an alternating optimization approach that updates the coding by first identifying an active set of features and then optimizing only on that set. The running time of each iteration is linear to the number of instances, therefore could be slow for large scale problems. Mairal et al. (2010a) suggested an online learning algorithm that updates the dictionary by processing instances in sequence. Their empirical results show a significant running time decrease.
We introduce here a new primal-dual approach for matrix completion that scales well to large and especially sparse matrices. Empirical results illustrate the effectiveness of our method in comparison to recently proposed alternatives. We then extend the approach to NMF and dictionary learning for sparse coding. Both of the two problems suffer non-convex objective functions. Following the trace norm, we introduce new norms of matrices that implicitly induce non-negativity, sparsity or both. By combining the norms with loss functions, we derive the new optimization problems that are convex relaxations to the original. The primal-dual approach can be applied to solve these problems with slightly different subproblems of finding the most violated constraints.

3.1 Derivation of the dual

It can be shown that the trace norm of matrix $X$ with factorization $X = UV^T$ is equivalent to (e.g. Srebro et al., 2005)

$$
\|X\|_* = \min_{UV^T = X} \frac{1}{2} (\|U\|_F^2 + \|V\|_F^2) \tag{3.1}
$$

where $U$ and $V$ need not be low rank so that the factorization always exists. Consider then an extended symmetric matrix

$$
W = \begin{pmatrix} UU^T & UV^T \\ VU^T & VV^T \end{pmatrix} \tag{3.2}
$$

whose upper right and lower left components equal $X$ and $X^T$, respectively. As a result, $\|X\|_* = \min_{W_{UR} = X} \text{tr}(W)/2$ where $W^{UR}$ is the upper right part of $W$. By construction, $W$ is symmetric and positive semi-definite.
We can also expand the observations into a symmetric matrix

\[ Z = \begin{pmatrix} 0 & Y \\ Y^T & 0 \end{pmatrix} \quad (3.3) \]

and use \( \Omega \) as the index set to identify observed entries in the upper right and lower left components of \( Z \). Formally, \( \Omega = \{(i, j) | 1 \leq i, j \leq m+n, (i, j-n) \text{ or } (i-n, j) \in D\} \).

With these definitions, the primal trace norm regularization problem is equivalent to the extended problem

\[
\min_{W \in S} \sum_{(i,j) \in \Omega} \text{Loss}(Z_{i,j} - W_{i,j}) + \lambda \text{tr}(W) \quad (3.4)
\]

where \( S \) is the cone of positive semi-definite matrices in \( \mathbb{R}^{(m+n) \times (m+n)} \).

We introduce dual variables for each observed entry in \( Z \) via Legendre conjugate transformations of the loss functions.

\[
\text{Loss}(z) = \max_q \{qz - \text{Loss}^*(q)\} \quad (3.5)
\]

The Lagrangian involving both primal and dual variables is then given by

\[
\mathcal{L}(A, X, Z) = \sum_{(i,j) \in \Omega} \left[ A_{i,j} (Z_{i,j} - X_{i,j}) - \text{Loss}^*(A_{i,j}) \right] \\
+ \lambda \text{tr}(X) \quad (3.6)
\]

where \( A \) can be written as

\[
A = \begin{pmatrix} 0 & Q \\ Q^T & 0 \end{pmatrix} \quad (3.7)
\]

and \( Q \) is sparse such that \( Q_{i,j} = 0 \) is \( (i, j) \notin D \). To introduce the dual variables for the positive semi-definite constraint, we consider the dual cone of \( S \) which is defined
\[ S^* = \{ E \in \mathbb{R}^{(m+n)\times(m+n)}, \text{tr}(E^T M) \geq 0, \forall M \in S \} \] (3.8)

\( S \) is self-dual so that \( S^* = S \). The Lagrangian is then

\[ \mathcal{L}(A, E, X, Z) = \sum_{(i,j) \in \Omega} \left[ A_{i,j}(Z_{i,j} - X_{i,j}) - \text{Loss}^*(A_{i,j}) \right] \]

\[ + \lambda \text{tr}(X) - \text{tr}(E^T X) \] (3.9)

To solve for \( X \), we set \( d/X \mathcal{L}(A, E, X, Z) = 0 \), and get

\[ \lambda I - A = E \in S \] (3.10)

Inserting the solution back into the Lagrangian, we obtain

\[ L(A) = \sum_{(i,j) \in \Omega} A_{i,j}Z_{i,j} - \text{Loss}^*(A_{i,j}) \] (3.11)

Since \( E \) does not show up in the Lagrangian, we can replace the equation (3.10) with a constraint \( \lambda I - A \in S \). The formulation can be simplified by considering the original \( Q \) and \( Y \) which correspond to the upper right components of \( A \) and \( Z \). The dual problem in these variables is then

\[
\begin{align*}
\text{maximize} & \quad \text{tr}(Q^T Y) - \sum_{(i,j) \in D} \text{Loss}^*(Q_{i,j}) \\
\text{subject to} & \quad \lambda^2 I - Q^T Q \in S
\end{align*}
\] (3.12)
3.2 Solving the dual

There are three challenges with the dual. The first one is the separation problem, i.e., finding vectors $b$, $\|b\| = 1$, such that $\|Qb\|^2 > \lambda^2$, where $Q$ refers to the current solution. Each such $b$ can be found efficiently, precisely because $Q$ is sparse. The second challenge is effectively solving the dual under a few spectral constraints. For this, we derive a new block coordinate descent approach (cf. Tseng, 1993). The third challenge concerns the problem of reconstructing the primal matrix $W$ from the dual solution. By including only a few spectral constraints in the dual, we obtain a low-rank primal solution. We can thus explicitly maintain a primal-dual pair of the relaxed problem with fewer constraints throughout the optimization.

**The separation problem.** We will iteratively add constraints represented by $b$. The separation problem we must solve is then: given the current solution $Q$, find $b$ for which $\|Qb\|^2 > \lambda^2$. This is easily solved by finding the eigenvector of $Q^TQ$ with the largest eigenvalue. For example, the power method

$$
 b = \text{randn}(m, 1). \text{ Iterate } b \leftarrow Q^TQb, \ b \leftarrow b/\|b\| \quad (3.13)
$$

is particularly effective with sparse matrices. If $\|Qb\|^2 > \lambda^2$ for the resulting $b$, then we add a single constraint $\|Qb\|^2 \leq \lambda^2$ into the dual. Note that $b$ does not have to be solved exactly; any $b$ provides a valid constraint albeit not necessarily the tightest. An existing constraint can be easily tightened later on with a few iterations of the power method, starting with the current $b$. We can fold this tightening together with the block coordinate optimization discussed below.

**Primal-dual block coordinate descent.** The second problem is to solve the dual subject to $\|Qb^l\|^2 \leq \lambda^2$, $l = 1, \ldots, k$, instead of the full set of constraints $Q^TQ \leq \lambda^2 I$. This partially constrained dual problem can be written as

$$
 \text{tr}(Q^TY) - \sum_{(i,j) \in D} \text{Loss}^*(Q_{i,j}) - \sum_{l=1}^{k} h(\|Qb^l\|^2 - \lambda^2) \quad (3.14)
$$

40
where \( h(z) = \infty \) if \( z > 0 \) and \( h(z) = 0 \) otherwise. The advantage of this form is that each \( h(\|Qb\|^2 - \lambda^2) \) term is a convex function of \( Q \) (a convex non-decreasing function of a convex quadratic function of \( Q \), and therefore itself convex). We can thus obtain its conjugate dual as

\[
h(\|Qb\|^2 - \lambda^2) = \sup_{\xi \geq 0} \left\{ \xi (\|Qb\|^2 - \lambda^2)/2 \right\}
\]

\[
= \sup_{\xi \geq 0, v} \left\{ v^T Qb - \|v\|^2/(2\xi) - \xi \lambda^2/2 \right\}
\]

where the latter form is jointly concave in \((\xi, v)\) where \( b \) is assumed fixed. This step lies at the core of our algorithm. By relaxing the supremum over \((\xi, v)\), we obtain a linear, not quadratic, function of \( Q \). The new Lagrangian is given by

\[
\mathcal{L}(Q, V, \xi) = \text{tr}(Q^T Y) - \sum_{(i,j) \in D} \text{Loss}^*(Q_{i,j})
\]

\[
- \sum_{l=1}^k \left[ (v^l)^T Q b^l - \|v^l\|^2/(2\xi^l) - \xi^l \lambda^2/2 \right]
\]

\[
= \text{tr}(Q^T (Y - \sum_l v^l (b^l)^T))
\]

\[
- \sum_{(i,j) \in D} \text{Loss}^*(Q_{i,j}) + \sum_{l=1}^k \left[ \frac{\|v^l\|^2}{2\xi^l} + \frac{\xi^l \lambda^2}{2} \right]
\]

which can be maximized with respect to \( Q \) for fixed \((\xi^l, v^l), l = 1, \ldots, k\). Functionally, our primal-dual algorithm seeks to iteratively minimize \( \mathcal{L}(V, \xi) = \max_Q \mathcal{L}(Q, V, \xi) \) while explicitly maintaining \( Q = Q(V, \xi) \). Note also that by maximizing over \( Q \), we reconstitute the loss terms \( \text{tr}(Q^T(Y - X)) - \sum_{(i,j) \in D} \text{Loss}^*(Q_{i,j}) \). The predicted rank \( k \) matrix \( W \) is therefore obtained explicitly as \( X = \sum_l v^l (b^l)^T \). By allowing \( k \) constraints in the dual, we search over rank \( k \) predictions.

The iterative algorithm proceeds by selecting one \( l \), fixing \((\xi^j, v^j), j \neq l\), and optimizing \( \mathcal{L}(V, \xi) \) with respect to \((\xi^l, v^l)\). As a result, \( \mathcal{L}(V, \xi) \) is monotonically decreasing. Let \( \hat{X} = \sum_{j \neq l} v^j (b^j)^T \), where only the observed entries need to be evaluated. We consider two variants for minimizing \( \mathcal{L}(V, \xi) \) with respect to \((\xi^l, v^l)\):
**Method 1:** When the loss function is not strictly convex, we first solve $\xi^l$ as function of $v^l$ resulting in $\xi^l = \|v^l\|/\lambda$. Recall that $\mathcal{L}(V, \xi)$ involves a maximization over $Q$ that recreates the loss terms with a predicted matrix $X = \tilde{X} + v^l(b^l)^T$. By dropping terms not depending on $v^l$, the relevant part of $\min_{\xi^l} \mathcal{L}(V, \xi)$ is given by

$$
\sum_{(i,j) \in D} \text{Loss}(Y_{i,j} - \tilde{X}_{i,j} - v^l_{u} b^l_{i}) + \lambda \|v^l\| \tag{3.15}
$$

which is a simple primal group Lasso minimization problem over $v^l$. Since $b^l$ is fixed, the problem is convex and can be solved by standard methods.

**Method 2:** When the loss function is strictly convex, we can first solve $v^l = \xi^l Q b^l$ and explicitly maintain the maximization $Q = Q(\xi^l)$. The minimization problem over $\xi^l \geq 0$ is

$$
\max_Q \left\{ \text{tr}(Q^T(Y - \tilde{X})) - \sum_{(i,j) \in D} \text{Loss}^*(Q_{i,j}) - \xi^l(\|Q b^l\|^2 - \lambda^2)/2 \right\} \tag{3.16}
$$

where we have dropped all the terms that remain constant during the iterative step. The optimization is decomposable for each row of $Q$,

$$
\max_{Q_{u,I_u}} \left\{ Q_{u,I_u}(Y_{u,I_u} - X_{u,I_u})^T - \sum_{i \in I_u} \text{Loss}^*(Q_{i,j}) - \xi^l Q_{u,I_u} b^l_{i} (b^l_{i})^T Q_{u,I_u}^T \right\} \tag{3.17}
$$

where $I_u = \{i : (i, j) \in D\}$ is the index set of observed elements for a row (user) $u$. For example, the squared loss, $Q(\xi^l)$ is obtained in closed form:

$$
Q_{u,I_u}(\xi^l) = (Y_{u,I_u} - \tilde{X}_{u,I_u}) \left(1 - \frac{\xi^l b^l_{I_u} (b^l_{I_u})^T}{1 + \xi^l \|b^l_{I_u}\|^2}\right), \tag{3.18}
$$

In general, an iterative solution is required. Subsequently, the optimal value $\xi^l \geq 0$ is set as follows. If $\|Q(0)b^l\|^2 \leq \lambda^2$, then $\xi^l = 0$. Otherwise, since $\|Q(\xi^l)b^l\|^2$ is monotonically decreasing as a function of $\xi^l$, we find (e.g., via bracketing) $\xi^l > 0$ such
that \( \|Q(\xi^l)b^l\|^2 = \lambda^2 \).

### 3.3 Constraints and convergence

The algorithm described earlier monotonically decreases \( \mathcal{L}(V, \xi) \) for any fixed set of constraints corresponding to \( b^l, l = 1, \ldots, k \). Any additional constraint further decreases this function. We consider here how quickly the solution approaches the dual optimum as new constraints are added. To this end, we write our algorithm more generally as minimizing

\[
F(B) = \max_Q \{ \text{tr}(Q^TY) - \text{Loss}^*(Q_{i,j}) \\
+ 1/2\text{tr}((\lambda^2 I - Q^TQ)B) \}
\]  

(3.19)

where \( B \) is a positive semi-definite \( m \times m \) matrix. The solution of the maximization problem is denoted as \( Q(B) \). For any fixed set of constraints, our algorithm minimizes \( F(B) \) over the cone \( B = \sum_{l=1}^k \xi^l b^l(b^l)^T \) that corresponds to constraints \( \text{tr}((\lambda^2 I - Q^TQ)b^l(b^l)^T) > 0, \forall l \). \( F(B) \) is clearly convex as a point-wise supremum of linear functions of \( B \). We further constrain \( \text{tr}(B) \leq C \).

**Lemma 3.3.1.** Assume the loss function \( \text{Loss}(z) \) is strictly convex with a Lipschitz continuous derivative (e.g., the squared loss), and \( |d(\text{Loss}(z))| \to \infty \) only if \( |z| \to \infty \), then \( F(B) \) has a Lipschitz continuous derivative. (see Appendix for proof)

We denote \( B^r \) as the cone created by the constraints from the primal dual algorithm at step \( r \) and \( B^* \) as the minimizer of \( F(B) \). The objective is monotonic decreasing because we add one constraint at each iteration. For the convergence rate, we have the following theorem.

**Theorem 3.3.2.** Under the assumptions discussed above, \( F(B^r) - F(B^*) = O(1/r) \). (see Appendix for proof)
3.4 Non-negative matrix factorization

Non-negative matrix factorization (NMF) separates a matrix into a product of two non-negative matrices. The problem is typically written as

$$\min_{U, V} \sum_{(i,j) \in D} (Y_{i,j} - (UV^T)_{i,j})^2$$

(3.20)

where $U \in \mathbb{R}^{n \times k}$ and $V \in \mathbb{R}^{m \times k}$ are both matrices with nonnegative entries. In general, the optimization problem defined in this way is non-convex and NP-hard (Vavasis, 2009). Lee and Seung (2001) describes a multiplicative update method that revises the decomposition into $U$ and $V$ iteratively while keeping $U$ and $V$ nonnegative,

$$V_{i,a} = V_{i,a} \frac{(Y^T U)_{i,a}}{(V U^T U)_{i,a}}, \quad U_{u,a} = U_{u,a} \frac{(Y V)_{u,a}}{(U V^T V)_{u,a}}$$

(3.21)

The updates monotonically decrease the objective and thus are guaranteed to converge. Another method for solving NMF is a projected gradient algorithm (Berry et al., 2006). It updates in the negative of the gradient direction, and projects the solution back to non-negative space,

$$U = \text{proj}(U - \epsilon_u \partial f / \partial U), \quad V = \text{proj}(V - \epsilon_v \partial f / \partial V)$$

(3.22)

where $f$ is the objective function, $\epsilon_u$ and $\epsilon_v$ are the step sizes. The projection operation is basically $\text{proj}(x) = \max(0, x)$.

The algorithms above are guaranteed to converge, however due to non-convexity of objective, it may converge to a local optimum. A variety of NMF with different constraints have been studied. Ding et al. (2010) proposed Semi-NMF that relaxes the non-negativity constraint for one of the factor matrices, and Covex-NMF that further constrained a factor matrix to be a convex combination of input data. These variations create more structure in the problem and achieve better results than NMF in some situations. However, despite the variations of constraints, there is still the
possibility of multiple local optima.

As before, we look for convex relaxations of this problem via trace norm regularization. However, this step requires some care. The set of matrices $UV^T$, where $U$ and $V$ are non-negative and of rank $k$, is not the same as the set of rank $k$ matrices with non-negative elements. The difference is well-known even if not fully characterized.

Let us first introduce some basic concepts such as completely positive and co-positive matrices. A matrix $W$ is completely positive if there exists a nonnegative matrix $B$ such that $W = BB^T$. It can be seen from the definition that each completely positive matrix is also positive semi-definite. A matrix $C$ is co-positive if for any $v \geq 0$, i.e., a vector with non-negative entries, $v^T Cv \geq 0$. Unlike completely positive matrices, co-positive matrices may be indefinite. We will denote the set of completely positive symmetric matrices and co-positive matrices as $S^+$ and $C^+$, respectively. $C^+$ is the dual cone of $S^+$, i.e., $(S^+)^* = C^+$. (the dimensions of these cones are clear from context).

Following the derivation of the dual in section 3, we consider expanded symmetric matrices $W$ and $Z$ in $\mathbb{R}^{(m+n) \times (m+n)}$, defined as before. Instead of requiring $W$ to be positive semi-definite, for NMF we constrain $W$ to be a completely positive matrix, i.e., in $S^+$. The primal optimization problem can be given as

$$\min_{W \in S^+} \sum_{(i,j) \in \Omega} \text{Loss}(Z_{i,j} - W_{i,j}) + \lambda \text{tr}(W) \quad (3.23)$$

Notice that the primal problem involves infinite constraints corresponding to $W \in S^+$ and is difficult to solve directly.

We start with the Lagrangian involving primal and dual variables:

$$\mathcal{L}(A, C, W, Z) = \sum_{(i,j) \in \Omega} \left[ A_{i,j} (Z_{i,j} - W_{i,j}) - \text{Loss}^*(A_{i,j}) \right]$$

$$+ \lambda \text{tr}(W) - \text{tr}(C^TW) \quad (3.24)$$
where \( C \in \mathcal{C}^+ \) and \( A = [0, Q; Q^T, 0] \) as before. By setting \( d/W \mathcal{L}(A, E, W, Z) = 0 \), we get

\[
\lambda I - A = C \in \mathcal{C}^+
\] (3.25)

Substituting the result back into the Lagrangian, the dual problem becomes

\[
\begin{align*}
\text{maximize} & \quad tr(A^T Z) - \sum_{(i,j)\in\Omega} \text{Loss}^*(A) \\
\text{subject to} & \quad \lambda I - A \in \mathcal{C}^+
\end{align*}
\] (3.26)

where \( A \) is a sparse matrix in the sense that \( A_{i,j} = 0 \) if \((i, j) \notin \Omega \). The co-positivity constraint is equivalent to \( v^T(\lambda I - A)v \geq 0, \forall v \geq 0 \).

Similarly to the primal dual algorithm, in each iteration \( v^l = [a^l; b^l] \) is selected as the vector that violates the constraint the most. This vector can be found by solving

\[
\max_{v \geq 0, \|v\| \leq 1} v^T Av = \max_{a, b \geq 0, \|a\|^2 + \|b\|^2 \leq 1} a^T Qb
\] (3.27)

where \( Q \) is the matrix of dual variables (before expansion), and \( a \in \mathbb{R}^m, b \in \mathbb{R}^n \).

While the sub-problem of finding the most violating constraint is NP-hard in general, it is unnecessary to find the global optimum. Any non-negative \( v \) that violates the constraint \( v^T(\lambda I - A)v \geq 0 \) can be used to improve the current solution. At the optimum, \( \|a\|^2 = \|b\|^2 = 1/2 \), and thus it is equivalent to solve

\[
\max_{a, b \geq 0, \|a\|^2 = \|b\|^2 = 1/2} a^T Qb
\] (3.28)

A local optimum can be found by alternatingly optimizing \( a \) and \( b \) according to

\[
\begin{align*}
a &= h(Qb)/ (\sqrt{2}\|h(Qb)\|) \\
b &= h(Q^Ta)/ (\sqrt{2}\|h(Q^Ta)\|)
\end{align*}
\] (3.29)
where \( h(x) = \max(0, x) \) is the element-wise non-negative projection. The running time across the two operations is directly proportional to the number of observed entries. The stationary point \( v_s = [a_s; b_s] \) satisfies \( \|v_s\| = 1 \) and \( Av_s + \max(0, -Av_s) + \|h(Av_s)\|v_s = 0 \) which are necessary (but not sufficient) optimality conditions for (3.27). As a result, many elements in \( a \) and \( b \) are exactly zero so that the decomposition is not only non-negative but sparse.

Given \( v^l, l = 1, \ldots, k \), the Lagrangian is then

\[
L(A, \xi) = tr(A^T Z) - \sum_{(i,j) \in \Omega} \text{Loss}^*(A) - \sum_l \xi^l ((v^l)^T Av^l - \lambda)
\]

where \( \xi^l \geq 0 \) and \( \sum_{l=1}^k \xi^l v^l (v^l)^T \in S^+ \).

To update \( \xi^l \), we fix \( \xi^j (j \neq l) \) and optimize the objective with respect to \( A \) and \( \xi^l \),

\[
\max_{A, \xi^l \geq 0} \quad tr(A^T (Z - \sum_{j \neq l} \xi^j v^j (v^j)^T)) - \sum_{(i,j) \in \Omega} \text{Loss}^*(A) - \xi^l ((v^l)^T Av^l - \lambda)
\]

It can be seen from the above formulation that our primal solution without the \( l^{th} \) constraint can be reconstructed from the dual as \( \tilde{X} = \sum_{j \neq l} \xi^j v^j (v^j)^T \in S^+ \). Only a small fraction of entries of this matrix (those corresponding to observations) need to be evaluated. If the loss function is the squared loss, then the variables \( A \) and \( \xi^l \) have closed form expressions. Specifically, for a fixed \( \xi^l \),

\[
A(\xi^l)_{i,j} = Z_{i,j} - \tilde{W}_{i,j} - \xi^l v^l_i v^l_j
\]  

(3.30)

If \( (v^l)^T A(0)v^l \leq \lambda \), then the optimum \( \hat{\xi}^l = 0 \). Otherwise

\[
\hat{\xi}^l = \frac{\sum_{(i,j) \in \Omega} (Z_{i,j} - \tilde{W}_{i,j}) v^l_i v^l_j - \lambda}{\sum_{(i,j) \in \Omega} (v^l_i v^l_j)^2}
\]  

(3.31)
Notice that the updates for NMF are different from general MF in section 3. In order to preserve non-negativity, we fix \( v^l = [a^l, b^l] \) as constant while previously only \( b^l \) is fixed and \( a^l \) is optimized to get a tighter constraint. If we consider the semi-NMF variation such that only one factor is required to be non-negative, then the same update rules as before can be applied by fixing only the non-negative component and optimizing the other.

The update of \( A \) and \( \xi^l \) takes linear time in the number of observed elements. If we update all \( \xi^l \) \((l = 1, \ldots, k)\) in each iteration, then the running time is \( k \) times longer that is the same as the update time in multiplicative update methods (Lee and Seung, 2001) that aim to directly solve the optimization problem (3.20). Because the objective is non-convex, the algorithm gets frequently trapped in locally optimal solutions. It is also likely to convergence slowly, and often requires a proper initialization scheme (Boutsidis and Gallopoulos, 2008). In contrast, the primal-dual method discussed here solves a convex optimization problem. With a large enough \( \lambda \), only a few constraints are necessary, resulting in a low rank solution. The key difficulty in our approach is identifying a violating constraint. The simple iterative method may fail even though a violating constraint exists. This is where randomization is helpful in our approach. We initialize \( a \) and \( b \) randomly and run the separation algorithm several times so as to get a reasonable guarantee for finding a violated constraints when they exist.

### 3.5 Sparse matrix factorization

Dictionary learning for sparse coding seeks to factorize a matrix into a normalized matrix (dictionary) and a sparse matrix (coding) (Mairal et al., 2010b; Lee et al., 2007). The problem is given by:

\[
\min_{U, V} \text{Loss}(Y - UV^T) + \lambda |V|_1 \\
\text{s.t.} \quad \|U_{:,i}\|_2 \leq 1, \forall i
\]
where $Y \in \mathbb{R}^{m \times n}$, $U \in \mathbb{R}^{m \times d}$ and $V \in \mathbb{R}^{n \times d}$. The problem is non-convex and therefore has a local optimum issue. To obtain a convex relaxation to sparse coding, we first define a matrix norm

$$
\|X\|_f = \min_{U \in \mathbb{R}^{n \times \infty}, c \in \mathbb{R}^{m \times \infty}} |W|_1
$$

$$s.t. \quad X = UV^T, \quad \|U_{i,:}\|_2 \leq 1, \forall i
$$

The fact that $\|X\|_f$ is a norm can be shown by $\|0\|_f = 0$, $\|cX\|_f = |c|\|X\|_f$, and triangle inequality $\|X + Y\|_f \leq \|X\|_f + \|Y\|_f$. Given $\|X\|_f$, we can relax the rank constraint in sparse coding, and obtain a convex formulation

$$\min_X \quad \text{Loss}(Y - X) + \lambda \|X\|_f \quad (3.32)$$

Solving the primal optimization problem directly is difficult. Instead, we consider the dual problem. The dual of $\|X\|_f$ is given by

$$
\|E\|_f^* = \max(\text{tr}(E^TX), \quad s.t. \quad \|E\|_f = 1)
$$

$$= \max(\text{tr}(E^T(u^Tv)), \quad s.t. \quad \|u\|_2 \leq 1, |v|_1 = 1)
$$

$$= \max((v^TE^TEv)^{1/2}, \quad s.t. \quad |v| = 1)
$$

In our formulation, the dual variables arise from a Legendre conjugate transformation of a loss function:

$$\text{Loss}(Y - X) = \max_Q(\text{tr}(Q^T(Y - X)) - \text{Loss}^*(Q)) \quad (3.33)$$

The Lagrangian is then given by

$$\mathcal{L}(Q, E, X) = \text{tr}(Q^T(Y - X)) - \text{Loss}^*(Q) + \lambda \text{tr}(X^TE) \quad s.t. \quad \|E\|_f^* \leq 1 \quad (3.34)$$
To solve for $X$, we set $d/XL(Q, Z, X) = 0$, and get $Q = \lambda Z$. Substituting the solution back, we obtain the dual

$$\max_Q \quad \text{tr}(Q^T Y) - \text{Loss}^*(Q) \quad s.t. \quad \|Q\|_f^* \leq \lambda$$

(3.35)

or equivalently,

$$\max_Q \quad \text{tr}(Q^T Y) - \text{Loss}^*(Q) \quad s.t. \quad v^T Q^T Qv \leq \lambda^2, \forall |v|_1 \leq 1$$

(3.36)

It turns out that the constraint is decomposable such that

$$\max \quad v^T Q^T Qv = \max_{|v| \leq 1} \|Q(:, i)\|^2$$

(3.37)

The matrix norm $\|X\|_f$ is equivalent to

$$\|X\|_f = (\|X\|_f^*)^* = \sum_{i=1}^n \|X(i, :)\|_2 = \|X\|_{2,1}$$

(3.38)

The primal reduces to a group lasso problem and each row of $X$ can be learned independently. The optimum solution of the codeword matrix $V$ is simply an indicator matrix, i.e. $V_{i,j} = 0$ if $i \neq j$. Clearly, the rank constraint in original sparse coding problem plays an essential role in making instances interplay.

To control the rank of solution, we introduce additional regularization similar to trace norm in the dual.

$$\|E\|_{f(\beta)}^* = \max(\|Ev\|, \ s.t. \ |v| + \beta \|v\| = 1)$$

(3.39)

The $L_2$ norm $\|v\|$ is a group sparsity penalization that favors $v$ to be a zero vector, thus resulting low rank primal solutions.
We now consider a slightly different format of the constraints that is easier to handle,

\[
\max_{v} \left| v \right| + \beta \left\| v \right\| = 1 \quad \iff \quad \max_{\|v\|=1} \left\| Qv \right\| - \lambda |v| \leq \lambda \beta \quad (3.40)
\]

Since \( \|Qv\| = \max_{\|u\|=1} u^T Qv \), we solve the sub-problem by alternatingly optimizing \( u \) and \( v \),

\[
u = Qv / \|Qv\|,
\]

\[
\tilde{v} = S_\lambda(Q^T u),
\]

\[
v = \tilde{v} / \|\tilde{v}\|. \quad (3.41)
\]

where \( S_\lambda(x) = \text{sign}(x) \max(|x| - \lambda, 0) \) is the soft-threshold operator. Given selected constraints \((u^l, v^l), l = 1, ..., k\), we solve the dual subject to these constraints via a coordinate descent algorithm. The Lagrangian involves the dual variable \( Q, (u^l, v^l) \) and conjugate dual of the constraints are

\[
\mathcal{L}(Q, U, V, \xi) = \text{tr}(Q^T Y) - \text{Loss}^*(Q) - \sum_{l=1}^{k} \xi_l ((u^l)^T Q v^l - \lambda |v^l| - \lambda \beta)
\]

\[
= \text{tr}(Q^T (Y - \sum_{l=1}^{k} \xi_l u^l (v^l)^T)) - \text{Loss}^*(Q) + \sum_{l=1}^{k} \lambda \xi_l (|v^l| + \beta). \quad (3.42)
\]

Each iteration we optimize one \( \xi^l \) while fixing the rest.

We compare the primal-dual algorithm to the greedy algorithm that selects rank-1 component at each iteration by solving

\[
\min_{u, v} \text{Loss}(Y - \tilde{W} - uv^T) + \lambda |v| + \lambda \beta \|v\| \quad (3.43)
\]

This can be solved by alternatingly optimizing \( u \) and \( v \). Optimizing \( v \) when \( u \) is fixed requires solving a linear regression problem with \( L_1 + L_2 \) regularization. In comparison, the corresponding update step for \( v \) in (3.41) has a closed form, therefore it is more efficient.
3.6 Independent Component Analysis

In this section, we consider a variation of ICA (Le et al., 2011). Similarly to sparse coding, we assume observations $y$ are generated from a set of sources $x$: $y = Ax$, or equivalently $x = A^{-1}y = Wy$. In addition, ICA assumes that the sources are independent, i.e. $E[x_ix_j] = 0$ for $i \neq j$. To obtain unique solutions, usually $E[x_ix_i] = 1$ is also required.

Before solving ICA, the data matrix $Y \in \mathbb{R}^{m \times n}$ where $m$ is the number of variables and $n$ is the number of instances are whitened such that $E[yy^T] = YY^T/n = I$. Under the independence assumption, we obtain,


(3.44)

We are interested in a sparse solution of $x$ in the sense that only a few sources are active for each instance. One common approach is to minimize the $L1$ norm of $X = WY$, so the optimization problem is given by

$$\min_W |WY|$$

$$s.t. \; WW^T = I$$

(3.45)

Solving the problem directly is difficult since its objective function is non-smooth and projection of $W$ to the constrained area is expensive. We consider relaxing the equivalence constraint via squared loss,

$$\min_W \|WW^T - I\|^2 + \lambda|WY|$$

(3.46)

When $\lambda \rightarrow 0$, the above two optimization problems are equivalent. Notice that the regularization $|WY|$ is non-decomposable for $W$, so it is improper to use many efficient algorithms for $L1$ regularized problems. This motivates us to modify the
optimization problem to be solely based on $X = WY$. Since $Y$ is whitened, we have

$$\|X^TX - Y^TY\|^2 = \|Y^TW^TWY - Y^TY\|^2 = \text{tr}(Y^T(W^TW - I)YY^T(W^TW - I)Y)$$
$$= n^2\|W^TW - I\|^2 = n^2(\|WW^T - I\|^2 - d + m)$$ (3.47)

We can therefore replace $\|WW^T - I\|^2$ by $\|Y^T - X^TX\|^2$ which yields

$$\min_X \|Y^TY - X^TX\|^2 + \lambda |X|$$ (3.48)

To solve the problem efficiently, we consider its dual problem. As above, we can define a matrix norm of semi-definite matrix

$$\|W\|_g = \min_{X^TX=W} |X|$$ (3.49)

and its dual norm

$$\|W\|_g^* = \max_{|v|\leq 1} v^TWv = \max_i W_{i,i}$$ (3.50)

The dual norm $\|W\|_g^*$ is very similar to $\|W\|_f^*$, but $\|W\|_g^*$ and is defined over the cone of semi-definite matrices. Unlike $\|W\|_f$, $\|W\|_g = (\|W_g\|)^*$ is hard to evaluate. The dual problem is given by

$$\max_Q \text{tr}(Q^TY^TY) - \text{tr}(Q^TQ)/2$$
$$\text{s.t. } v^TQv - \lambda |v| \leq 0, \forall \|v\|^2 \leq 1$$ (3.51)

where $Q \in \mathbb{R}^{n \times n}$. The subproblem of finding the most violated constraint can be solved by alternatingly optimizing $u$ and $v$,

$$u = Qv,$$
$$\tilde{v} = S_\lambda(u),$$
$$v = \tilde{v}/\|\tilde{v}\|.$$
One potential issue is that the size of $Q$ can be very large. To remedy this, we only keep the decomposition form of $Q = Y^T Y - \sum \xi_i v_i (v_i)^T$. The running time of computing $Qv$ is $O(mn)$.

From the dual, we can see that ICA and sparse coding solve very similar problems. The major difference is that sparse coding intends to reconstruct the original instance while ICA intends to preserve the inner product between instances.

3.7 Experiments

3.7.1 Primal-dual algorithm for matrix completion

There are many variations of our primal-dual (PD) method pertaining to how the constraints are enforced and when they are updated. We introduced our method as one that fully optimizes all $\xi^l$ corresponding to the available constraints prior to searching for additional constraints. Another variant, analogous to gradient based methods, is to update only the last $\xi^k$ associated with the new constraint without ever going back and iteratively optimizing any of the previous $\xi^l (l < k)$ (or the constraints themselves). This method adds constraints faster as $\xi^l$ are updated only once when introduced. Another variant is to update all the previous $\xi^l$ together with their constraint vectors $b^l$ (through the power method) before introducing a new constraint. By updating $b^l$ as well, this approach replaces the previous constraints with tighter ones. This protocol can reduce the number of constraints that are added and is thus useful in cases where a very low rank solution is desirable.

Here we compare the two variants of our method to recent approaches proposed in the literature. We begin with the state of art method proposed in (Jaggi and Sulovsk, 2010) which we refer to as JS. Similar to PD, JS formulates a trace norm regularization problem using extended matrices $X, Z \in \mathbb{R}^{(m+n) \times (m+n)}$ which are derived from the predictions and observed data. It provides a solution for the following optimization
problem,

\[
\min_{X \geq 0, \tr(X)=t} \text{Loss}(X, Z) \quad (3.52)
\]

Instead of penalizing the trace of \(X\), JS fixes \(\tr(X)\) to a constant \(t\). In each iteration, JS finds the maximum eigenvector of the gradient, \(v^k\), and updates \(X\) according to \(X = (1-t_k)X + t_k v^k(v^k)^T\) where \(t_k\) is an optimized step size. During the optimization process, the trace norm constraint is always satisfied. Though JS and PD attempt to optimize different objective functions, they will converge to the same solution in the limit if we set \(t = tr(X^*)\) where \(X^*\) is the optimum matrix from our regularized objective (2.5).

The comparison is on Movielens 10M dataset which contain \(10^7\) ratings of 69878 users on 10677 movies. Following the setup in (Jaggi and Sulovsk, 2010), we use partition \(r_b\) provided with the dataset which has 10 test ratings per user. The regularization parameter \(\lambda\) in the PD method is set to 50. For the JS method, the corresponding value of the constant is \(t = tr(X^*) = 30786\) where \(X^*\) is the optimum from the PD method that fully optimizes all \(\xi_l\) before searching for additional constraints.

In order to ensure that we perform a fair comparison with JS, rather than updating \(\xi_l\) for all \(l = 1, 2, ..., k\) at iteration \(k\), only \(\xi^k\) is updated (the first variant introduced above). In other words, all \(\xi^l\) are updated only once when the corresponding constraint is incorporated. Figure 3-1a compares the test RMSE of JS and PD as a function of time. The test error of PD decreases much faster than JS. Figure 3-1b compares the test error as a function of rank, i.e., as a function of iteration rather than time. PD performs significantly better than JS when rank is small at the beginning owing to the better optimization step. In Figure 3-2a, the regularized primal objective function values \(J(W) = L(W) + \lambda\|W\|_*\) are compared as a function of time. Note that the objectives used by the methods are different and thus JS does not directly optimize this function. However, it will converge to the same limiting value at the optimum based on how the parameters \(t\) and \(\lambda\) are set. From the figure,
\( J(W) \) does not converge to the optimum over the training set as the limiting values should agree. This is also evident in Figure 3-2b which shows the training error as a function of time. PD error is consistently lower than JS.

We also compare our PD method to a recently proposed GECO algorithm (see Shalev-Shwartz et al., 2011). GECO seeks to solve the rank constrained optimization problem

\[
\min_{\text{rank}(X) \leq r} \text{Loss}(X, Y) \quad (3.53)
\]

It maintains a decomposition in the form \( X = UV^T \) and, at each iteration, increases the rank of \( U \) and \( V \) by concatenating vectors \( u \) and \( v \) that correspond to the largest singular values of the gradient of \( \text{Loss}(X, Y) \). Moreover, it optimizes the expanded
$U$ and $V$ by rotating and scaling the corresponding vectors. This solidifying step is computationally expensive but nevertheless improves the fixed-rank result. In order to fairly compare with their method, we use the second variant of our method. In other words, before adding a new constraint in iteration $k$, we update all $\xi^l$ including $b^l$ for $l = 1, 2, \ldots k$ for a fixed number of times related to their algorithm ($q$ times). Note that our algorithm here is different from that used in comparison to JS. Due to the complexity of GECO, we used a smaller dataset, Movielens 1M, which contains $10^6$ ratings of 6040 users regarding 3706 movies. Following the setup in (Shalev-Shwartz et al., 2011), 50% of the ratings are used for training, and the rest for testing.

Figure 3-3 compares the test RMSE of GECO and PD. GECO becomes very slow when the rank increases (the complexity can be $O(r^6)$ as a function of rank $r$). In light of the fact that it took 17 hours to produce rank 9 solutions from GECO, we did not attempt to use it on larger datasets. Figure 3-b compares the test RMSE as a function of rank. PD is substantially faster and needs fewer constraints to perform well.

### 3.7.2 Non-negative matrix factorization

Next, we test the scalability of our algorithm for Non-negative Matrix Factorization and its performance on language modeling problem that estimates the probability of...
observing a word given the previous one. It can be formulated as a matrix problem with rows as the previous word and columns as the next word. The potential issue is that the resulting model may be over-fitting to the trained documents. Here we want to test how well our algorithm for NMF generalize to other documents.

The experiment is on 20news - group data which contains short documents separated into 20 groups. To test the generalization ability, we pick the documents in category 'talk.politics.guns' as training data, and 'rec.sport.baseball' as test data. The number of documents in training and test data are 910 and 993 respectively. To clean the dataset, we removed the words that appeared less than five times in all documents. We also removed a list of common stop words. The resulting dataset contains \( n = 7242 \) distinct words. We then generate an \( n \times n \) rating matrix \( R \) such that \( R_{i,j} \) is the frequency that word \( i \) followed by word \( j \) is presented in the training data. Since some bigrams in the test data do not appear in the training data, we add a pseudo-count \( \nu \) to each entry, and the resulting matrix is \( R' = R + \nu 11^T \).

Notice the matrix is not sparse anymore, however the primal-dual algorithm can still be efficient due to the simple structure of \( R' \). The dual variable in fining the most violated constraint at step \( k \) is \( Q = R + \nu 11^T - \sum_{l=1}^{k} \xi^l U_{.,l} V_{.,l}^T \) and

\[
Qb = Rb + \nu \sum_{i} b_i 1 - \sum_{l=1}^{k} \xi^l U_{.,l} V_{.,l}^T b
\]
\[
Q^T a = R^T a + \nu \sum_{i} a_i 1 - \sum_{l=1}^{k} \xi^l V_{.,l} U_{.,l}^T a
\] (3.54)

The matrix \( R \) is sparse and \( k \) is usually small, and is therefore efficient to compute. The other change is that the update of \( x_i^l \) now becomes

\[
\hat{\xi}^l = \frac{\sum_{(i,j) \in D} R_{i,j} U_{i,l} V_{j,l} + \nu (\sum_i U_{i,l})(\sum_j V_{j,l}) - \sum_{p=0}^{l-1} \sum_{i,j} U_{i,l} V_{j,l} U_{i,p} V_{j,p} - \lambda/2}{(\sum_i (U_{i,l})^2)(\sum_j (V_{j,l})^2)}
\] (3.55)

where \( D \) is the set of non-zeros in the \( R \).

After obtaining \( \xi^l, U \) and \( V \), we normalize \( V_{j,l} \) such that \( \sum_j V_{j,l} = 1 \) and \( \xi^l = \ldots \)
\( \xi^l \sum_j V_{j,l} \), \( \forall l \). Then \( V_{j,l} \) can be seen as a distribution over words, and NMF as a mixture model with probability distribution,

\[
Pr(i, j) = \frac{\sum_l \xi^l U_{i,l} V_{j,l}}{\sum_l \xi^l U_{i,l}}
\]

where \( Pr(i, j) \) is the probability of observing word \( j \) after word \( i \), and \( \xi^l U_{i,l} \) serves as the weight of component \( l \).

The number of non-zeros elements in the training data matrix \( R \) is 35321 or equivalently 0.07% of the entire elements. We compare the test data log-likelihood of NMF to the bigram model that relies on statistics i.e. \( Pr(i, j) = R'_{i,j} / \sum_k R'_{i,k} \). We try the value \( \nu \) for the bigram method from the set \{0.0001, 0.001, 0.01, 0.1, 1\} and select \( \nu = 0.01 \) that obtains the largest log-likelihood. Next we use \( \nu = 0.01 \) to train the NMF model and compare the result. The regularization \( lambda \) has a minor effect in the result, and is set to be \( \lambda = 0.1 \).

The results are shown in Figure 4. The log-likelihood of NMF grows up quickly at the beginning and surpass the Bigram baseline with rank around 20. Comparing to the size of the problem \( n = 7042 \), we conclude NMF is able to capture the bigram relationship with a few ranks. The log-likelihood becomes close to being flat with over 200 rank.

Figures 5 plots the log-likelihood as a function of time and also the time as a function of rank. The total running time is 432 seconds on a 2 x 2.66 GHz desktop. The majority of running the time results from the computation introduced by \( \sum_{l=1}^k \xi^l U_{i,l} V_{j,l}^T \), which is linear to the rank \( k \). The total running time therefore grows quadratic with rank as shown in the right figure. Due to the fact that each iteration is very efficient and we are able to find very good constraints, the log-likelihood grows very quickly at the beginning and surpasses the Bigram baseline in only a few seconds.
3.7.3 Sparse matrix factorization

We apply the algorithm in section 3.5 for sparse coding to RS. In the experiment, the sparsity regularization is only implemented on the item side such that the resulting factor $V$ will be sparse. In addition to $L_1$ regularization as in (3.39) (trace norm with $L_1$ regularization), we also consider $L_0$ as an alternative (trace norm with $L_0$ regularization). We use theoretical storage space of $V$ after compression to measure the effect of sparsity. Assuming the value of entries follow a Gamma distribution $\mathcal{G}(\alpha, \sigma)$ where $\sigma^2$ is estimated as the variance of $V$. To store a value $x$ up to $\delta$
accuracy would require \( \log_2(G(x|0, \sigma)\delta) \) bits because the cumulative probability of the range \( [x - \delta/2, x + \delta/2] \) can be approximated as \( G(x|0, \sigma)\delta \). In addition, the number of nonzero elements \( k \) requires \( \log(m) \) bits and their positions need \( \log(C_k^m) \) bits. Adding all the components together will result in our estimation of the total storage of \( V \) after compression.

Figure 3-6 compares the test RMSE of the regular trace norm, trace norm with \( L_0 \) and with \( L_1 \) regularization as a function of storage bytes. The experiment uses the same training and test data split as above. The regularization parameter \( \lambda \) is set to 50 throughout the experiment. The sparsity parameter \( \beta \) is 300 for \( L_0 \) and \( \beta = 15 \) for \( L_1 \) constraints. The values of these parameters are selected by cross validation and the results are not very sensitive to them. The proportion of non-zero entries with \( L_0 \) constraints is \( \beta/m \approx 3\% \), and approximately 10\% with \( L_1 \) constraints. The figure shows that sparsity induced norms have a clear advantage of compression in the collaborative filtering tasks.

Figure 3-6: test RMSE comparison of Trace norm, Trace norm w. \( L_0 \), Trace norm w. \( L_1 \) regularization
Chapter 4

Privacy mechanism

Matrix completion accuracy depends directly on the amount of information that each user is willing to share with the system (Alvim et al., 2012). It may be possible in some cases to avoid this statistical trade-off by building Peer-to-Peer networks with homomorphic encryption which is computationally challenging (Canny, 2002; Miller et al., 2004). We aim to address the statistical question directly.

The statistical trade-off between accuracy and privacy depends on the notion of privacy we adopt. A commonly used privacy concept is Differential Privacy (DP) (Dwork, 2008), first introduced to protect information leaked from database queries. In a recommender context, users may agree to a trusted party to hold and aggregate their data, and perform computations on their behalf. Privacy guarantees are then demanded for any results published beyond the trusted party (including back to the users). In this setting, differential privacy can be achieved through obfuscation (adding noise) without a significant loss of accuracy (McSherry and Mironov, 2009).

In contrast to McSherry and Mironov (2009), we view the system as an untrusted entity, and assume that users wish to guard their own data. We depart from differential privacy and separate computations that can be done locally (privately) by individual users and computations that must be performed by the system (e.g., aggregation).
For example, in terms of low rank matrices, only the item features have to be solved by the system. The corresponding user features can be obtained locally by the users and subsequently used for ranking.

From this perspective, we divide the set of users into two pools, the set of public users who openly share their preferences, and the larger set of private users who require explicit privacy guarantees. We show theoretically and demonstrate empirically that a moderate number of public users is sufficient for an accurate estimation of item features. The remaining private users can make use of these item features without any release of information. Moreover, we propose a new second order privacy concept which uses limited (second order) information from the private users as well, and illustrate how recommendations can be further improved while still maintaining marginal deniability of private information.

4.1 Problem formulation and summary of results

Figure 4-1 illustrates the distributed system setup. The users are separated into public users and private users. Each user holds his own data. The server sends the current estimate of item factor matrix $V$ to each user and receives a completed rating vector in return.

**Recommender setup without privacy** To simplify the analysis, we need to scale the objective slightly differently. The basic estimation problem, without any privacy considerations, is then given by

$$\min_{X \in \mathbb{R}^{m \times n}} \frac{1}{N} \sum_{(i,j) \in \Omega} (Y_{i,j} - X_{i,j})^2 + \frac{\lambda}{\sqrt{mn}} \|X\|_*$$

(4.1)

where $N = |\Omega|$ is the total number of observed ratings. The factor $\sqrt{mn}$ ensures that the regularization does not grow with either dimension.
The above formulation requires the server to explicitly obtain predictions for each user, i.e., solve for $X$. We can instead write $X = UV^T$ and $\Sigma = (1/\sqrt{mn})VV^T$, and solve for $\Sigma$ only. If the server then communicates the resulting low rank $\Sigma$ (or just $V$) to each user, the users can reconstruct the relevant part of $U$ locally, and reproduce $X$ as it pertains to them. To this end, let $\phi_i = \{ j : (i, j) \in \Omega \}$ be the set of observed entries for user $i$, and let $Y_{i,\phi_i}$ be a column vector of user $i$’s ratings. Then we can show that Eq.(4.1) is equivalent to solving

$$
\min_{\Sigma \in S^+} \sum_{i=1}^{n} Y_{i,\phi_i}^T (\lambda' I + \Sigma_{\phi_i,\phi_i}) Y_{i,\phi_i} + \sqrt{nm} \| \Sigma \|_*
$$

(4.2)

where $S^+$ is the set of positive semi-definite $m \times m$ matrices and $\lambda' = \lambda N/\sqrt{nm}$. By solving $\hat{\Sigma}$, we can predict ratings for unobserved items (index set $\phi_i^c$ for user $i$) by

$$
\hat{X}_{i,\phi_i^c} = \Sigma_{\phi_i^c,\phi_i} (\lambda' I + \Sigma_{\phi_i,\phi_i})^{-1} Y_{i,\phi_i}
$$

(4.3)

Note that we have yet to address any privacy concerns. The solution to Eq.(4.2) still requires access to full ratings $Y_{i,\phi_i}$ for each user.

**Recommender setup with privacy** Our privacy setup assumes an untrusted server. Any user interested in guarding their data must therefore keep and process
their data locally, releasing information to the server only in a controlled manner. We will initially divide users into two broad categories: public and private. Public users are willing to share all their data with the server while private users are unwilling to share any. This strict division is removed later when we permit private users to release, in a controlled manner, limited information pertaining to their ratings (second order information) so as to improve recommendations.

Any data made available to the server enables the server to model the collective experience of users. For example, to solve Eq.(4.2), we will initially consider the setting where Eq.(4.2) is solved on the basis of public users only. We use an EM type algorithm for training. In the E-step, the current $\Sigma$ is sent to public users to complete their rating vectors and sent back to the server. In the M-step, $\Sigma$ is then updated based on these full rating vectors. The resulting $\hat{\Sigma}$ (or $\hat{V}$) can be subsequently shared with the private users, enabling the private users (their devices) to locally rank candidate items without releasing any private information. The estimation of $\hat{\Sigma}$ is then improved by asking private users to share second order relational information about their ratings without any release of marginal selections/ratings.

Note that we do not consider privacy beyond ratings. In other words, we omit any subsequent release of information due to users exploring items recommended to them. Figure 4-2 summarizes the operations conducted on user and server sides.

**Summary of results** We outline here our major results towards characterizing the role of public users and the additional controlled release of information from private users.

1) We show that $\hat{\Sigma} = \sqrt{X^TX/\sqrt{nm}}$ can be estimated in a consistent, accurate manner on the basis of public users alone. In particular, we express the error $\|\hat{\Sigma} - \Sigma\|_F$ as a function of the total number of observations. Moreover, if the underlying public user ratings can be thought of as i.i.d. samples, we also bound $\|\hat{\Sigma} - \Sigma^*\|_F$ in terms of the number of public users. Here $\Sigma^*$ is the true limiting estimate. See section 4.2.1
2) We show how the accuracy of predicted ratings $\hat{X}_{i,\phi_i}$ for private users relate to the accuracy of estimating $\Sigma$ (primarily from public users). Since the ratings for user $i$ may not be related to the subspace that $\Sigma$ lies in, we can only characterize the accuracy when sufficient overlap exists. We quantify this overlap, and show how $\|\hat{X}_{i,\phi_i} - \hat{X}_{i,\phi_i}\|$ depends on this overlap, accuracy of $\Sigma$, and the observation noise. See section 4.2.2 for details.

3) Having established the accuracy of predictions based on public users alone, we go one step further and introduce a new privacy mechanism and algorithms for gleaning additional relational, second order, information from private users. This second order information is readily used by the server to estimate $\Sigma$. The privacy concept constructively maintains first order (marginal) deniability for private users. We demonstrate empirically the gains from the additional second order information. See section 4.3.
4.2 Analysis

4.2.1 Statistical Consistency of $\hat{\Sigma}$

Let $\hat{X}$ be a solution to Eq.(4.1). We can write $\hat{X} = \hat{U}\hat{V}^T$, where $\hat{U}^T\hat{U} = \hat{I}_m$ with 0/1 diagonal. Since $\hat{\Sigma} = \frac{1}{\sqrt{mn}}\sqrt{\hat{X}^T\hat{X}}$ we can first analyze errors in $\hat{X}$ and then relate them to $\hat{\Sigma}$. To this end, we follow the restricted strong convexity (RSC) analysis (Negahban et al., 2012). However, their result depends on the inverse of the minimum number of ratings of all users and items. In practice (see below), the number of ratings decays exponentially across sorted users, making such a result loose. We provide a modified analysis that depends only on the total number of observations $N$.

Throughout the analysis, we assume that each row vector $\hat{X}_{i\cdot}$ belongs to a fixed $r$ dimensional subspace. We also assume that both noiseless and noisy entries are bounded, i.e. $|Y_{i,j}|, |\hat{X}_{i,j}| \leq \alpha, \forall (i,j)$. For brevity, we use $\|Y - X\|_{\Omega}^2$ to denote the empirical loss $\sum_{(i,j)\in\Omega}(Y_{i,j} - X_{i,j})^2$. The restricted strong convexity property (RSC) assumes that there exists a constant $\kappa > 0$ such that

$$\frac{\kappa}{mn}\|\hat{X} - \hat{\Phi}\|^2_F \leq \frac{1}{N}\|\hat{X} - \hat{\Phi}\|_{\Omega}^2$$

for $\hat{X} - \hat{\Phi}$ in a certain subset. RSC provides the step from approximating observations to approximating the full underlying matrix. It is satisfied with high probability provided that $N = (m + n)\log(m + n))$.

Assume the SVD of $\hat{X} = \hat{P}\hat{S}\hat{Q}^T$, and let $\text{row}(X)$ and $\text{col}(X)$ denote the row and column spaces of $X$. We define the following two sets,

$$A(P, Q) := \{X, \text{row}(X) \subseteq \hat{P}, \text{col}(X) \subseteq \hat{Q}\}$$

$$B(P, Q) := \{X, \text{row}(X) \subseteq \hat{P}^\perp, \text{col}(X) \subseteq \hat{Q}^\perp\}$$

Let $\pi_A(X)$ and $\pi_B(X)$ be the projection of $X$ onto sets $A$ and $B$, respectively, and
\( \pi_A = I - \pi_A \), \( \pi_B = I - \pi_B \). Let \( \Delta = \hat{X} - \hat{X} \) be the difference between the estimated and the underlying rating matrices. Our first lemma demonstrates that \( \Delta \) lies primarily in a restricted subspace and the second one guarantees that the noise remains bounded.

**Lemma 4.2.1.** Assume \( \epsilon_{i,j} \) for \( (i,j) \in \Omega \) are i.i.d. sub-gaussian with \( \sigma = \| \epsilon_{i,j} \|_{\psi_1} \). Then with probability \( 1 - \frac{c}{N^{2\epsilon_r}} \), \( \| \pi_B(\Delta) \|_* \leq \| \pi_B(\Delta) \|_* + \frac{2c^2\sigma^2\sqrt{mn}}{N\lambda} \log^2 N \). Here \( h > 0 \) is an absolute constant associated with the sub-gaussian noise.

If \( \lambda = \lambda_0 c \sigma \frac{\log^2 N}{\sqrt{N}} \), then \( \frac{c^2\sigma^2\sqrt{mn}}{N\lambda} = \frac{c\sigma \log N}{\lambda_0} \sqrt{\frac{mn}{N}} = b \log N \sqrt{\frac{m}{N}} \) where we leave the dependence on \( n \) explicit. Let \( D(b, n, N) \) denote the set of difference matrices that satisfy lemma 4.2.1 above. By combining the lemma and the RSC property, we obtain the following theorem.

**Theorem 4.2.2.** Assume RSC for the set \( D(b, n, N) \) with parameter \( \kappa > 0 \) where \( b = \frac{c\sigma \sqrt{m}}{\lambda_0} \). Let \( \lambda = \lambda_0 c \sigma \frac{\log N}{\sqrt{N}} \), then we have \( \frac{1}{\sqrt{mn}} \| \Delta \|_F \leq 2c\sigma \left( \frac{1}{\sqrt{\kappa}} + \frac{\sqrt{r}}{\kappa} \right) \log N \sqrt{\frac{m}{N}} \) with probability at least \( 1 - \frac{c}{N^{2\epsilon_r}} \), where \( h, c > 0 \) are constants. (See Appendix for proof)

The bound in the theorem consists of two terms, pertaining to the noise and the regularization. In contrast to Negahban et al. (2012), the terms only relate to the total number of observations \( N \).

We now turn our focus on the accuracy of \( \hat{\Sigma} \). First, we map the accuracy of \( \hat{X} \) to that of \( \hat{\Sigma} \) using a perturbation bound for polar decomposition (Mathias, 1997).

**Lemma 4.2.3.** If \( \frac{1}{\sqrt{mn}} \| \hat{X} - \hat{X} \|_F \leq \delta \), then \( \| \hat{\Sigma} - \Sigma \|_F \leq \sqrt{2}\delta \)

This completes our analysis in terms of recovering \( \hat{\Sigma} \) for a fixed size underlying matrix \( \hat{X} \). As a final step, we turn to the question of how the estimation error changes as the number of users or \( n \) grows. Let \( \hat{X}_i \) be the underlying rating vector for user \( i \) and define \( \Theta^\circ = \frac{1}{mn} \sum_{i=1}^n \hat{X}_i^T \hat{X}_i \). Then \( \hat{\Sigma} = (\Theta^\circ)^\frac{1}{2} \). If \( \Theta^* \) is the limit of \( \Theta^\circ \), then \( \Sigma^* = (\Theta^*)^\frac{1}{2} \). We bound the distance between \( \hat{\Sigma} \) and \( \Sigma^* \).

**Theorem 4.2.4.** Assume \( \hat{X}_i \) are i.i.d samples from a distribution with support only in a subspace of dimension \( r \) and a bounded norm of \( \| \hat{X}_i \| \leq \alpha \sqrt{m} \). Let \( \beta_1 \) and \( \beta_r \) be
the smallest and largest eigenvalues of \( \Sigma^* \). Then, for large enough \( n \), with probability at least \( 1 - \frac{r}{n^2} \),

\[
\| \hat{\Sigma} - \Sigma^* \|_F \leq 2\sqrt{r} \alpha \sqrt{\frac{\beta_1 \log n}{\beta_1 n}} + o(\frac{\log n}{n}) \tag{4.6}
\]

Combining the two theorems and using triangle inequality, we obtain a high probability bound on \( \| \hat{\Sigma} - \Sigma^* \|_F \). Assuming the number of ratings for each user is larger than \( \xi m \), then \( N > \xi nm \) and the bound grows in the rate of \( \eta (\log n / \sqrt{n}) \) with \( \eta \) being a constant that depends on \( \xi \). For large enough \( \xi \), the required \( n \) to achieve a certain error bound is small. Therefore a few public users with a large number of ratings could be enough to obtain a good estimate of \( \Sigma^* \).

### 4.2.2 Prediction accuracy

It is now possible to characterize the error in the predicted ratings \( \hat{X}_{i,\phi} \) for all users as defined in Eq.(4.3). For brevity, we use \( \delta \) to represent the bound on \( \| \hat{\Sigma} - \Sigma^* \|_F \) obtained on the basis of our results above. We also use \( x_\phi \) and \( x_{\phi^c} \) as shorthand for \( X_{i,\phi} \) and \( X_{i,\phi^c} \) with the understanding that \( x_\phi \) typically refers to a new private user.

The important issue for us here is that the partial rating vector \( x_\phi \) may be of limited use. For example, if the number of observed ratings is less than rank \( r \), then we would be unable to identify a rating vector in the \( r \) dimensional subspace, even with the absence noise. It is desired to disregard this limitation in our analysis by assuming that the observations have enough signal to be useful. Let SVD of \( \Sigma^* \) be \( Q^* S^* (Q^*)^T \), and \( \beta_1 \) be its minimum eigenvalue. We constrain the index set of observations \( \phi \) such that it belongs to the set

\[
D(\gamma) = \left\{ \phi \subseteq \{1, \ldots, m\}, s.t. \|x\|_F^2 \leq \gamma \frac{m}{|\phi|} \|x_\phi\|_F^2, \forall x \in \text{row}((Q^*)^T) \right\}
\]

The parameter \( \gamma \) depends on how the low dimensional sub-space is aligned with
the coordinate axes. We are only interested in characterizing prediction errors for observations that are in $\mathcal{D}(\gamma)$. This is quite different from the RSC property. Our main result is then

**Theorem 4.2.5.** Suppose $\|\Sigma - \Sigma^*\|_F \leq \delta \ll \beta_1$, $\phi \in \mathcal{D}(\gamma)$. For any $\hat{x} \in \text{row}((Q^*)^T)$, our observation $x_\phi = \hat{x}_\phi + \epsilon_\phi$ where $\epsilon_\phi \sim \text{Sub}(\sigma^2)$ is the noise vector. The predicted ratings over the remaining entries are given by $\hat{x}_{\phi^c} = \Sigma_{\phi^c,\phi}(\lambda I + \Sigma_{\phi,\phi})^{-1}x_\phi$. Then, with probability at least $1 - \exp(-c_2 \min(c_1^4, \sqrt{|\phi|c_1^2}))$,

$$\|x_{\phi^c} - \hat{x}_{\phi^c}\|_F \leq 2\sqrt{\lambda} + \delta(\sqrt{\frac{m}{|\phi|}} + 1)(\frac{\|\hat{x}\|_F}{\sqrt{\beta_1}} + \frac{2c_1 \sigma |\phi|^{\frac{1}{4}}}{\sqrt{\lambda}})$$

where $c_1, c_2 > 0$ are constants. (See Appendix for proof)

The upper bound contains two terms. The term proportional to $\|\hat{x}\|_F/\sqrt{\beta_1}$ is due to the estimation error of $\Sigma^*$, while the term proportional to $2c_1 \sigma |\phi|^{\frac{1}{4}}/\sqrt{\lambda}$ is a consequence of the noise in the observed ratings. All the proofs are provided in the supplementary material.

### 4.3 Controlled privacy for private users

The theoretical results have demonstrated that a relatively small sample size of public users with many ratings suffices for a reasonable performance guarantee for both public and private users. Empirical results, addressed in the next section, support this claim. However, since public users enjoy no privacy guarantees, it is preferred to limit the required number of such users by requesting private users to contribute in a limited manner while maintaining specific notions of privacy.

**Definition 4.3.1.** Privacy preservation mechanism. Let $\mathcal{M} : \mathbb{R}^{m \times 1} \rightarrow \mathbb{R}^{m \times 1}$ be a random mechanism that takes a rating vector $r$ as input and outputs $\mathcal{M}(r)$ of the same dimension with $j^{th}$ element $\mathcal{M}(r)_j$. We say that $\mathcal{M}(r)$ is element-wise, privacy
preserving if \( \Pr(\mathcal{M}(r)_j = z) = p(z) \) for \( j = 1, \ldots, m \), and for some fixed distribution \( p \).

For example, a privacy preserving mechanism \( \mathcal{M}(r) \) is element-wise private if its coordinates follow the same marginal distribution such as the uniform distribution. Note that such a mechanism can still output information about how different ratings interact via covariance, which is necessary for estimation.

**Discrete values.** Assume that each element in \( r \) and \( \mathcal{M}(r) \) belongs to a discrete set \( S \) with \( |S| = K \). A natural privacy constraint is to insist that the marginal distribution of \( \mathcal{M}(r)_j \) is uniform, i.e., \( \Pr(\mathcal{M}(r)_j = z) = 1/K \), for \( z \in S \). A trivial mechanism that satisfies the privacy constraint is to select each value uniformly at random from \( S \). In this case, the returned rating vector does not contribute to the server model. Our goal is to design a mechanism that preserves useful second order information.

We first assume that a small number of public user profiles are available, from which we can learn an initial model parameterized by \((\mu, V)\), where \( \mu \) is the item mean vector and \( V \) is a low rank component of \( \Sigma \). The server provides each private user the pair \((\mu, V)\) and asks once for a response \( \mathcal{M}(r) \). Here \( r \) is the user’s full rating vector, completed privately with the help of the server model \((\mu, V)\).

The mechanism \( \mathcal{M}(r) \) is assumed to be element-wise privacy preserving, thus releasing nothing about a single element in isolation. What information should it carry? If each user \( i \) provided their full rating vector \( r^i \), the server could estimate \( \Sigma \) according to \( \frac{1}{nm} (\sum_{i=1}^{n}(r^i - \mu)(r^i - \mu)^T)^{\frac{1}{2}} \). Thus, if \( \mathcal{M}(r) \) preserves the second order statistics to the extent possible, the server could still obtain an accurate estimate of \( \Sigma \).

Consider a particular user and their completed rating vector \( r \). Let \( P(x) = \Pr(\mathcal{M}(r) = x) \). We find the distribution \( P(x) \) by solving the following optimization problem geared towards preserving interactions between the ratings under the uniform marginal
where $K = |S|$. The exact solution is difficult to obtain due to the fact that the number of distinct assignments of $x$ is $K^m$. Instead, we consider an approximate solution. Let $x^1, ..., x^K \in \mathbb{R}^{m \times 1}$ be $K$ different vectors such that, for each $i$, $\{x_i^1, x_i^2, ..., x_i^K\}$ forms a permutation of $S$. If we choose $x$ with $\Pr(x = x^j) = 1/K$, then the marginal distribution of each element is uniform and therefore maintains element-wise privacy. It remains to optimize the set $x^1, ..., x^K$ in order to capture interactions between ratings.

A greedy coordinate descent algorithm is used to optimize $x^1, ..., x^K$. For each coordinate $i$, we randomly select a pair $x^p$ and $x^q$, and switch $x^p_i$ and $x^q_i$ if the objective function in (4.7) is reduced. The process is repeated a few times before moving on to the next coordinate. The algorithm can be implemented efficiently because each operation deals with only a single coordinate.

Finally, according to the mechanism, the private user selects one item from $x^j$, $j = 1, \ldots, K$, uniformly at random, and returns the discrete vector to the server. Because the resulting rating vectors from private users are noisy, the server decreases their weight relative to the information from public users as part of the overall M-step for estimating $\Sigma$.

Figure 4-3a is an example of $\mathcal{M}(r)$ for $S = \{1, 2, 3\}$. Three vectors $\gamma^1, \gamma^2, \gamma^3$ are randomly selected with the constraint that each of their coordinates is a permutation of $S$. Each coordinate system is then optimized sequentially until reaching convergence. The final output is chosen uniformly at random from $\{\gamma^1, \gamma^2, \gamma^3\}$.

**Continuous values.** Assuming the rating values are continuous and unbounded, we require instead that the returned rating vectors follow the marginal distributions with
a given mean and variance. Specifically, $E[\mathcal{M}(r)_i] = 0$ and $\text{Var}[\mathcal{M}(r)_i] = v$ where $v$ is a constant that remains to be determined. Note that once again any specific element of the returned vector will not, in isolation, carry any information specific to the element.

As before, we search for the distribution $P$ so as to minimize the $L_2$ error of the second order statistics under marginal constraints. For simplicity, denote $r' = r - \mu$ where $r$ is the true completed rating vector, and $u_i = \mathcal{M}(r)_i$. The objective is given by

$$\min_{P,v} E_{u \sim P} \| uu^T - r'r'^T \|_F^2$$

$$\text{s.t.} \quad E[u_i] = 0, \ \text{Var}[u_i] = v, \ \forall i. \quad (4.8)$$

Note that the formulation does not directly constrain that $P$ has identical marginals, only that the means and variances agree. The optimal solution does however and is shown next.

**Theorem 4.3.2.** Let $z_i = \text{sign}(r'_i)$ and $h = (\sum_{i=1}^{m} |r'_i|)/m$. The minimizing distribution of (4.8) is given by $\Pr(u = zh) = \Pr(u = -zh) = 1/2$.

The proof is left to the supplementary material. A few remarks are needed. The mechanism in this case is a two component mixture distribution, placing a probability

---

Figure 4-3: Example of $\mathcal{M}(r)$

(a) Discrete values with $S = \{1, 2, 3\}$

(b) Continuous values
mass on \( \text{sign}(r')h \) (vectorized) and \(-\text{sign}(r')h\) with equal probability. As a result, the server, knowing the algorithm that private users follow, can reconstruct \( \text{sign}(r') \) up to an overall randomly chosen sign. Note also that the value of \( h \) is computed from the user’s private rating vector and therefore releases some additional information about \( r' = r - \mu \), albeit minimally. To remove this information altogether, we could use the same \( h \) for all users and estimate it based on public users.

Figure 4-3b gives an example of privacy mechanism for continuous values with \( m = 2 \). The original vector is \( x \), and the output is \( zh \) or \(-zh\) both with probability \( 1/2 \). Notice that the output removes the magnitude of \( x \) on each coordinate, but still preserves the positive correlation among the two coordinates.

The privacy constraints will clearly have a negative impact on the prediction accuracy in comparison to having direct access to all the ratings. However, the goal is to improve accuracy beyond the information from public users alone by obtaining limited information from private users. While improvements are possible, the effects of limited information are evident in several ways. First, because private users provide no first order information, the estimation of mean rating values cannot be improved beyond that of public users. Second, the sampling method we use to enforce element-wise privacy adds noise to the aggregate second order information from which \( V \) is constructed. Finally, the server can run the M-step with respect to the private users’ information only once, whereas the original EM algorithm could entertain different completions for user ratings iteratively. Nevertheless, as illustrated in the next section, the algorithm can still achieve satisfiable accuracy, improving with each additional private user.

4.4 Experiments

We perform experiments on the Movielens 10M dataset which contains 10 million ratings from 69878 users on 10677 movies. The test set contains 10 ratings for each
user.

We begin by demonstrating that in fact a few public are sufficient for making accurate recommendations. Figure 4-4a shows the test performance of both weighted (Negahban et al., 2012) and unweighted (uniform) trace norm regularization as we add users. Here users with the most ratings are added first.

With only 1% of public users added, the test RMSE of unweighted trace norm regularization is 0.876 which is already close to the optimal prediction error. Note that the loss of weighted trace norm regularization actually begins to increase when the number of users increases. The reasoning is that the weighted trace norm penalizes less for users with fewer ratings. As a result, the resulting low dimensional subspace used for prediction is influenced more by users with fewer ratings.

The statistical convergence bound in section 4.2.1 involves both terms that decrease as a function of the number of ratings $N$ and the number of public users $n$. In fact, the two factors are usually coupled. It is however interesting to understand how they impact performance individually. Given a number of total ratings, we compare two different methods of selecting public users. In the first method, users with the most ratings are selected first, whereas the second method selects users uniformly at random. As a result, if we have same number of total number of ratings from each method, the second method selects many more users. Figure 4-4b shows that
the second method achieves better performance. An interpretation, based on the theory, is that the right side of the error bound (4.6) decreases as the number of users increases.

Figure 4-5: Test RMSE as a function of private user numbers. **PMC**: the privacy mechanism for continuous values; **PMD**: the privacy mechanism for discrete values; **Lap eps=1**: DP with Laplace noise, \( \epsilon = 1 \); **Lap eps=5**: same as before except \( \epsilon = 5 \); **SSLP eps=5**: sampling strategy described in (Duchi et al., 2013) with DP parameter \( \epsilon = 5 \); **Exact 2nd order**: with exact second order statistics from private users (not a valid privacy mechanism); **Full EM**: EM without any privacy protection.

We also show how performance improves based on controlled access to private user preferences. First, the top 100 users are taken whom have the most ratings corresponding to the ratings of the public users, and learn the initial prediction model from their ratings. To highlight possible performance gains, private users with more ratings are selected first. The results are similar if we select private users uniformly.

The rating values are from 0.5 to 5 with a total of 10 different discrete values. Following the privacy mechanism for discrete values, each private user generates ten different candidate vectors and returns one of them uniformly at random. In the M-step, the weight for each private user is set to 1/2 compared to 1 for public users. During training, after processing \( w = 20 \) private users, we update parameters \((\mu, V)\), recompute the rating vectors of public users, making predictions for the next batch
of private users more accurate. The privacy mechanism for continuous values is also tested under the same setup. We denote the two privacy mechanism as PMD and PMC, respectively.

We compare five different scenarios. First, we use a standard DP method that adds Laplace noise to the completed rating vector. Let the DP parameter be $\epsilon$, because the maximum difference between rating values is 4.5, the noise follows $Lap(0, 4.5/\epsilon)$. As before, we give a smaller weight to the noisy rating vectors and this is determined by cross validation.

Second, Duchi et al. (2012) proposed a notion of “local privacy” in which differential privacy is guaranteed for each user separately. An optimal strategy for a $d$-dimensional multinomial distribution in this case reduces effective samples from $n$ to $n\epsilon^2/d$ where $\epsilon$ is the DP parameter. In our case the dimension corresponds to the number of items making estimation challenging under DP constraints. We will also evaluate a comparison to this method and denote it as SSLP (sampling strategy for local privacy).

In addition, to understanding how our approximation to second order statistics affects the performance, we also compare to the case in which $r'a$ is given to the server directly where $a = \{-1, 1\}$ with probability $1/2$. In this way, the server can obtain the exact second order statistics using $r'r'T$. Note that this is not a valid privacy preserving mechanism. Finally, we compare to the case in which the algorithm can access private user rating vectors multiple times and update the parameters iteratively. Again, this is not a valid mechanism but illustrates how much could be gained.

Figure 4-5 shows the performance as a function of the number of private users. The standard Laplace noise method performs reasonably well when $\epsilon = 5$, however the corresponding privacy guarantee is very weak. SSLP improves the accuracy mildly. In contrast, with the privacy mechanism we defined in section 4.3, the test RMSE decreases significantly as more private users are added. If we use the exact second or-
der information without the sampling method, the final test RMSE would be reduced by 0.07 compared to PMD. Finally, full EM without privacy protection reduces the test RMSE by another 0.08. These performance gaps are expected because there is an inherent trade-off between accuracy and privacy.

4.5 Conclusion

Our contributions in this chapter are three-fold. First, we provided explicit guarantees for estimating item features in matrix completion problems. Second, we showed how the resulting estimates, if shared with new users, can be used to predict their ratings depending on the degree of overlap between their private ratings and the relevant item subspace. The empirical results demonstrate that even a small number of public users with a large number of ratings suffices for a good performance. Third, we introduced a new privacy mechanism for releasing second order information needed for estimating item features while maintaining 1st order deniability. The experiments show that this mechanism indeed performs well in comparison to other mechanisms. We believe that allowing different levels of privacy is an exciting research topic. An extension of our work would be to apply the privacy mechanism to the learning of graphical models in which second or higher order information plays an important role.
Chapter 5

Tensor factorization for set evaluation

Viewing recommender problems in terms of matrices is limiting. Indeed, several extensions have been proposed for aggregating multiple related completion problems (cf. multi-task learning, Goldberg et al. (2010); Singh and Gordon (2008)). When predictions are made in the presence of multiple dimensions, views, or relations, the problem is more naturally formulated as a multi-dimensional array or tensor. Indeed, any regression problem based on discrete features can be thought of as a tensor completion problem (Liu et al., 2009). A tensor is a direct generalization of a matrix and, as such, many matrix problems and algorithms can be generalized for tensors. Matrix completion problems involve highly incomplete matrices. The problem is even more prominent with tensors, highlighting the need for effective regularization methods.

Unlike matrices, many tensor related problems are NP-hard (Hillar and heng Lim, 2009). For example, the rank of a tensor is defined as the minimum number of rank-one components needed to reconstruct the tensor, and is NP-hard. Finding the best rank-one approximation to the tensor is also NP-hard. Despite NP-hardness, efficient algorithms have been proposed to approximately solve tensor related problems and have achieved excellent results (Acar et al., 2009, 2010; Kolda and Bader, 2009). However, the problem becomes more difficult when the number of modes increases. The
major difficulty in computation lies in the inherent nonlinearity of tensors. Therefore explicit control over the degree of nonlinearity is essential for tensor related optimization problems.

In this chapter, we focus on directly controlling the tensor factorization. A number of earlier approaches (see, e.g., Kolda and Bader (2009) for a review) have reduced tensor completion problems back into a set of matrix problems by unfolding the tensor across dimensions, and regularizing the rank of resulting matrices. Such methods permit the use of previously developed algorithms but can become infeasible when the number of dimensions increases. We develop an algorithm for tensor completion via rank-one tensor updates. A new convex regularization is used to simultaneously control the degree of nonlinearity, the order of the components in the sense of the number of non-uniform modes, as well as the number of rank-one components. As an illustrative example of tensor completion, we expand recommender problems to be over sets, i.e., predicting a set of items for each user. The empirical results show that our method outperforms competing methods. We will also study the consistency of our approach.

5.1 Notation

In this section, we introduce shorthand variables that will be used in the paper. Let $X \in \mathbb{R}^{m_1 \otimes m_2 \otimes \cdots \otimes m_n}$ be a n-way tensor with dimension $m_i$ for the $i^{th}$ mode. We define the inner product of two tensors as $\langle A, B \rangle = vec(A)^T vec(B)$ where $vec(\cdot)$ transforms a matrix into a row vector, and $\|A\|_2^2 = \langle A, A \rangle$. For simplicity, we denote $\beta_1 \otimes \beta_2 \otimes \ldots \beta_n$ as $\otimes_{j=1}^n \beta_j$. We also use $\beta(x)$ to denote the $x^{th}$ element of $\beta$.

The unfolding matrix $X(i) \in \mathbb{R}^{m_i \times \mathbb{R}^{\otimes j \neq i m_j}}$ is a transformation of tensor $X$ by making the $i^{th}$ mode as rows, and concatenating all other modes as columns.

Observations in a typical matrix recommender problem are tuples $(u, i, r)$ that represent users, items and ratings, respectively. These are observed entries of the underlying matrix $Y$ such that $Y_{u,i} = r$. $Y$ is usually very sparse in the sense that
many entries are missing. In order to reconstruct the missing entries, we can learn $X$, for example, by matching the observed entries while minimizing the trace (nuclear) norm.

Here we extend the setting to recommend a set of items to users. For example, a website can recommend a pair of ski boots, jacket, gloves and helmets to a user who enjoys skiing. In this case, each “item” now becomes a subset $S_i \subseteq \{1, 2, ..., n\}$ where $n$ is the total number of individual items. The number of items to recommend is now equal to the number of all subsets or $2^n$. It is inconceivable to expect to see even one observed rating per “item” in this case. A standard matrix factorization approach is no longer feasible.

We can obtain a more effective solution via tensor factorization. Let $b$ be a binary vector that encodes the selected subset: $b_k = 1$ if $k \in S_i$, and zero otherwise. The task now is to learn a tensor $X \in \mathbb{R}^{n \times \prod_{j=1}^{n} 2}$ with one mode for users and $n$ binary dimensional modes corresponding to individual items.

Extending the setup further, many prediction tasks involve categorical, namely more than binary, selections as inputs. We can also compile these into a tensor so that a general recommender problem involves learning a tensor $X \in \mathbb{R}^{n \times \prod_{i=1}^m}$ from partially observed data.

A Tensor can also be seen as a function $f : (a_1, a_2, ..., a_n) \rightarrow \mathbb{R}$ where $a_i$ is the index for mode $i$. Let $x_{j,k}[a]$ be a binary indicator whether $a$ equals $k$ for the $j^{th}$ mode. We can then represent the function $f(a_1, a_2, ..., a_n)$ as a linear combination of binary monomials $\prod_{j \in S} x_{j,k_j}[a_j]$, $S \subseteq \{1, 2, ..., n\}$, $k_j \in \{1, \ldots, m_j\}$. The number of such monomial terms is clearly exponential in $n$.

Intuitively, higher order monomials are uncommon, therefore we only have to use monomials with order less than or equal to $d$, i.e., $|S| \leq d$. With this constraint, the number of possible monomial terms reduces from exponential to polynomial $O(n^d)$. The resulting function $f$ admits a decomposition $f(a_1, \ldots, a_n) = \sum_{i=1}^{K} f_i(a_{S_i})$ where
$a_{S_i}$ is $\{a_j : j \in S_i\}$. In other words, each component depends only on a subset $S_i \subseteq \{1, \ldots, n\}$ of modes.

The set of all the monomials of order $d$ is still large. We will further collapse this set, or functionally tie them together, by representing components $f_i$ as rank one tensors:

$$X_i = \kappa(\otimes_{j \in S_i} \beta_j)(\otimes_{p \notin S} e_p)$$  \hspace{1cm} (5.1)

$$\|\beta_j\|_2 = 1, \ \|e_p\| = 1$$  \hspace{1cm} (5.2)

where $e_p$ for all modes $p$ is a uniform vector of norm one recalling that the dimension changes with mode. Note that even though we write the tensor loosely as $\kappa(\otimes_{j \in S_i} \beta_j)(\otimes_{p \notin S} e_p)$, the modes continue to be aligned with the sub-indexes. If $\{1\} \notin S$ then $e_1$ is associated with the first mode. Since each mode $e_p$, $p \notin S$, is a uniform vector, it has no effect other than overall scaling. In other words, mode $p$ is eliminated from the rank 1 component. The tensor with $K$ components is now given by

$$X = \sum_{i=1}^{K} \kappa^i(\otimes_{j \in S_i} \beta^i_j)(\otimes_{p \notin S_i} e_p)$$  \hspace{1cm} (5.3)

where $\|\beta^i_j\|_2 = 1$ for all $i = 1, \ldots, K$, $j \in S_i$.

### 5.2 Convex regularization

In this section, we consider a coherent regularization function for tensor decomposition that favors low order rank-one components. The regularization is proposed as an analog to the trace norm regularization for matrices.

First, it is known that the trace norm is equivalent to the following optimization
problem:
\[ \|X\|_* = \min_{\kappa, u, v} \sum_{i=1}^{\infty} |\kappa^i|, \quad s.t. \quad X = \sum_{i} \kappa^i u_i v_i^T. \] (5.4)

This property of the trace norm leads to a few rank-one update algorithms for solving optimization problems with trace norm regularization (Jaggi and Sulovsk, 2010; Dudik et al., 2012).

The analogous singular values of a tensor are NP-hard to compute. Therefore, in order to generalize the trace norm to tensor, we consider the second formulation that is based on an optimization problem.

Additionally, we are interested in recovering a tensor with low order components. Therefore, we consider a subspace of rank-ones tensors with order \(d\),

\[ \Phi_d := \{ \otimes_{j=1}^{n} \beta_j \mid \|\beta_j\|_2 = 1, \sum_j \delta(\beta_j \neq \frac{e}{\sqrt{m_j}}) = d \} \] (5.5)

where \(\delta(\cdot)\) is 1 if the statement inside is true. Parameter \(d\) controls the order of the rank-one component. Correspondingly, we can define a subspace of a tensor as

\[ \Psi_d := \{ X \mid X = \sum_{i} \kappa^i \otimes_{j=1}^{n} \beta^i_j, \quad \beta^i_j \in \cup_{k=1}^{d} \Phi_k \} \] (5.6)

In particular, for a tensor \(X \in \Psi_d\), we define the regularization as

\[ R_d(X) = \min_{\kappa, \beta} \sum_{i} |\kappa^i| \]

\[ s.t. \quad X = \sum_{i} \kappa^i \otimes_{j=1}^{n} \beta^i_j, \quad \otimes_{j=1}^{n} \beta^i_j \in \cup_{k=1}^{d} \Phi_k \]

As we discussed in previous sections, a low order rank-one tensor is favored as it reduces non-linearity in optimization.

Lemma 5.2.1. The regularization function \(R_d(X)\) is convex for \(X \in \Psi_d\).
Proof. It’s easy to see that \( R_d(cX) = c \cdot R_d(X) \). Let \( \{\kappa_1, \beta_1\} \) and \( \{\kappa_2, \beta_2\} \) be the solutions for minimization in (5.7) for \( X_1 \) and \( X_2 \), then \( \{(c_1 \kappa, (1 - c_1) \kappa_2), (\beta_1, \beta_2)\} \) satisfy the constraints in (5.7) for \( X = c_1 X_1 + (1 - c_1) X_2 \) with \( 0 \leq c \leq 1 \). Because \( R_d(X) \) is defined as the minimum of objective, we have

\[
R_d(cX_1 + (1 - c)X_2) \leq c \cdot R_d(X_1) + (1 - c) \cdot R_d(X_2) \tag{5.7}
\]

Lemma 5.2.2. Let \( A_{i,j} \in \mathbb{R}^{m_i \otimes m_j} \) be a matrix in mode \( i \) and \( j \), then for any \( X \in \Psi_d \),

\[
R_2(X) = \min_{A_{i,j}} \sum_{i,j: i \neq j} \sqrt{\prod_{k \neq i,j} m_k} \|A_{i,j}\|_*,
\]

subject to \( X = \sum_{i,j} A_{i,j} \otimes e^\otimes_{k \neq i,j} m_k \) \( (5.8) \)

which can be solved in polynomial time.

Proof. When \( d = 2 \) we can organize the rank-one components into groups according to their non-ones modes and let

\[
X_{i,j} = \sum_l \kappa_l^i \beta_l^j \otimes \beta_l^j \otimes \frac{e^\otimes_{k \neq i,j} m_k}{\sqrt{\prod_{k \neq i,j} m_k}} \tag{5.9}
\]

Where the term \( \sqrt{\prod_{k \neq i,j} m_k} \) is from the normalization constraint. Let \( A_{i,j} = \frac{\sum_l |\kappa_l^i| \beta_l^j}{\sqrt{\prod_{k \neq i,j} m_k}} \), then minimization over \( \sum_l |\kappa_l^i| \) would give \( \sqrt{\prod_{k \neq i,j} m_k} \|A_{i,j}\|_* \) from (5.4). Plugging it back to (5.7) yields the equation in the theorem.

The minimization problem associated with \( R_2(X) \) is convex. One approach to solve it is by iteratively optimizing one \( A_{i,j} \) while fixing the rest. Then the problem is reduced to minimizing \( \|A_{i,j}\|_* \) with linear constraints, which can be solved efficiently (Yang and Yuan, 2013).
Although many trace norm related problems are NP-hard to solve, $R_2(X)$ can be efficiently evaluated. With regularization, we assume the underlying tensor can be decomposed into a summation of components that are parameterized by matrices. Despite the fact that high order interactions among modes are ignored, the approach has achieved excellence performance (Rendle and Schmidt-Thieme, 2010a). However, previous work using explicit decomposition of low rank matrices shows that the optimization problem is non-convex. On the other hand, solving the problem with matrices $A_{i,j}$ has produced an issue of indefiniteness because multiple combinations of $A_{i,j}$ can give the same tensor. Directly solving with a tensor and using $R_2(X)$ regularization avoids the issue.

In general, evaluating $R_d(X)$ for $d > 2$ can be difficult in general. However, the value of $R_d(X)$ may be unnecessary. Later we will show that the way $R_d(X)$ is formulated leads to a natural rank-one tensor update algorithm. The regularization coefficients of $R_d(X)$ serves as a threshold for including a new component, and the algorithm can proceed without evaluating $R_d(X)$.

One interesting aspect of $R_d(X)$ is that its dual defined as

$$R^*_d(X) = \max_{Y \in \Psi_d} \langle Y, X \rangle, \quad s.t. \quad R_d(Y) \leq 1 \quad (5.10)$$

for any $X \in \Psi_d$. Remember that the dual of a trace norm is the max norm defined as the maximum singular value of matrices. Similarly, we have the following lemma,

**Lemma 5.2.3.** The dual of $R_d(X)$ for $X \in \Psi_d$ is given by

$$R^*_d(X) = \max_{\oplus_j \beta_j \in \bigcup_{k=1}^d \Phi_k} \langle X, \oplus_j \beta_j \rangle \quad (5.11)$$

**Proof.** $Y$ in equation (5.10) can be decomposed as $\sum_i \kappa^i \otimes_{j=1}^n \beta_j^i$ with $\beta_j^i \in \bigcup_{k=1}^d \Phi_k$ from the definition of $R_d(Y)$, then the objective is linear to $\kappa^i$ with constraint $\sum_i |\kappa^i| \leq 1$. The maximum is achieved when a single $\kappa$ is equal to 1, and the corresponding rank-one tensor has order less than or equal to $d$. \hfill \Box
The two functions $R_d(X)$ and $R_d^*(X)$ satisfy the duality inequality, for $X, Y \in \Psi_d$,

$$\langle X, Y \rangle \leq R_d(X)R_d^*(Y)$$ (5.12)

It can be proved directly via the definition of dual problems (5.10). For functions with different $d$’s, if $d_1 \leq d_2$ and $X \in \Psi_{d_1}$, then

$$R_{d_2}(X) \leq R_{d_1}(X), \quad R_{d_1}^*(X) \leq R_{d_2}^*(X)$$ (5.13)

The reason is that optimization problems for $R_{d_1}(X)$ and $R_{d_2}(X)$ optimize the same objective function while $R_{d_1}$ has more constraints. The same reason applies to the dual problem.

A high order tensor can be transformed into a matrix via unfolding. The row and column of the unfolded matrix $X^{(k)} \in R^{m_k} \otimes R^\otimes_{l:i\neq m,k}$ are mode $k$ and concatenation of all the rest of the modes respectively. From the construction of $R_d(X)$ and $R_d^*(X)$, if $X \in \Psi_d$ we have

$$\|X\|_2 \leq \|X^{(k)}\|_* \leq R_d(X), \quad R_d^*(X) \leq \|X^{(k)}\|_\infty \leq \|X\|_2$$

Compared to the lower bound, it is more difficult to get upper bound of $R_d(X)$. For $d = n$, we have following results,

**Lemma 5.2.4.** Let $X$ be a $n^{th}$ order tensor and $r_k = \text{rank}(X^{(k)})$ and without loss of generality assume $r_n$ is the largest besides $r_k$. Denote $\gamma = \prod_{i;i\neq m,k} r_i$, then we have the following inequalities,

$$R_n(X) \leq \gamma \|X^{(k)}\|_*, \quad \frac{1}{\gamma} \|X^{(k)}\|_\infty \leq R_n^*(X)$$

**Proof.** Assume the unfolded matrix has SVD as $X^{(k)} = \sum_i \kappa_i u_i \otimes v_i^T$ where $v_i$ is a normalized vector in space $R^\otimes_{l:i\neq m,k}$. $v_i$ can also be seen as a tensor. If we pick another mode $j$, unfold $v_i$ along the mode and compute its SVD, we get $v_i = \sum_j \theta_j^i p_j^i (Q_j^i)^T$. Now we have a set of components $(\kappa_i \theta_j^i) u_i \otimes \theta_j^i \otimes Q_j^i$. Notice that the first two modes
of each component are vectors. If we continue this process, eventually we will have a set of rank-one tensors. The trace norm has the following bounds,

\[ \|A\|_F \leq \|A\|_* \leq \sqrt{r} \|A\|_F \] (5.14)

where \( r \) is the rank of matrix \( A \). Since \( \theta^j_i \) are the singular values and \( \|v_i\|_2 = 1 \), we get \( \sum_j |\theta^j_i| \leq \sqrt{r} \). To eventually obtain rank-one tensors, we need \( n - 2 \) unfolding excluding the mode \( k \) and the last mode because the last mode already has a vector form. The set of rank-one tensors is a feasible point in the optimization problem (5.7), and the summation of coefficients gives an upper bound of \( R_n(X) \). By chaining the inequalities from each unfolding step and choosing mode \( n \) as the last mode, we get the scalar \( \gamma = \sqrt{r_{j_1}r_{j_2}...r_{j_{n-2}}} \) where \( j_i \) denotes the \( i^{th} \) mode for unfolding. Plugging it back to the definition of \( R_n(X) \) and \( \|X^{(k)}\|_* \), yields the inequality in the lemma. A similar proof also applies to obtaining the lower bound of \( R_n^*(X) \). The bound is tight when \( n = 2 \) because \( \gamma = 1 \) in this situation.

We want to emphasize that although different regularizations for tensors are usually related in terms of upper and lower bounds, they add distinct structures to the optimization problem and therefore result in very different solutions. For instance, trace norm regularization on \( \|X^{(k)}\|_* \) forces the matrix to have low rank. However, each vector in space \( \mathbb{R}^{\otimes j, j \neq k \neq m_j} \) is also a tensor, and without additional constraints it is not clear whether this tensor is generalizable. Another caveat is the dimension in \( \mathbb{R}^{\otimes j, j \neq k \neq m_j} \) grows exponentially with the number of modes.

### 5.3 Consistency guarantee

In this section, we analyze the statistical consistency of estimation with \( R_d(X) \) as regularization. Specifically, we consider a tensor completion problem whose goal is to predict missing entries given a set of observations. We denote the positions of observations in tensor as \( \Omega_i \in \mathbb{R}^{\otimes m_j} \) for \( i = 1, ..., N \), which has value 1 for observed
entry and 0 otherwise. The observation model is

\[ y_i = \langle \Omega_i, X \rangle + \epsilon_i \]  

(5.15)

where \( X \in \mathbb{R}^{\otimes j=1^{m_j}} \) and \( \epsilon_i \) is \( i.i.d. \) random noise. Assuming the noise follows a Gaussian distribution, we want to find the best \( X \in \Psi_d \) that approximates observations by solving

\[
\min_{X \in \Psi_d} \sum_{i=1}^{n} (y_i - \langle \Omega_i, X \rangle)^2 + \lambda R_d(X) 
\]

(5.16)

The overall objective function is convex, and therefore we can find its global optimum and analyze sample complexity statistically.

Let \( P_d(X) \) be the least square projection of tensor \( X \) to \( \Psi_d \), we want to find \( \hat{X} \in \Psi_d \) that is close to \( P_d(X^*) \) where \( X^* \) is the underlying tensor for estimation. We have ignored the components of \( X^* \) that have order higher than \( d \), and therefore the estimator can be biased. However, it is still valid if the high order part is small.

More specifically, in the following lemma, we bound the effect of noise and high order components in estimation.

For simplicity, we introduce \( M \in \mathbb{R}^{\otimes m_i} \) as the tensor for noise such that \( y_i = \langle \Omega_i, X + M \rangle \) and \( M \) is zero at non-observed positions. We also denote \( \| \Omega(X) \|_2^2 = \sum_i (\langle \Omega_i, X \rangle)^2 \) and \( \Delta = X^* - P_d(X^*) \). The following theorem bounds the empirical errors of the estimator.

\[ \| \Omega(\hat{X} - P_d(X^*)) \|_2^2 \leq 2\lambda R_d(P_d(X^*)) + \frac{\lambda^2}{16} \]  

(5.17)
Proof. Because \( \hat{X} \) achieves optimum, for any \( X \in \Psi_d \), we have

\[
\sum_{i=1}^{n} (y_i - \langle \Omega_i, \hat{X} \rangle)^2 + \lambda R_d(\hat{X}) \leq \sum_{i=1}^{n} (y_i - \langle \Omega_i, X \rangle)^2 + \lambda R_d(X) \tag{5.18}
\]

From our previous notation,

\[
y_i = \langle \Omega_i, P_d(X) + \Delta + M \rangle \tag{5.19}
\]

Replacing \( y_i \), and reorganizing terms yields

\[
\|\Omega(\hat{X} - P_d(X^*))\|_2^2 \leq \|\Omega(X - P_d(X^*))\|_2^2 + 2\langle X - \hat{X}, \Omega(M + \Delta) \rangle + \lambda(R_d(X) - R_d(\hat{X}))
\]

Given the assumption that \( \lambda \geq 2R_n(M) \) and \( \lambda \geq 2R_n(\Delta) \), we get

\[
\langle \hat{X} - X, \Omega(M + \Delta) \rangle \\
\leq (R_n^*(M) + R_n^*(\Omega(\Delta)))R_n(\hat{X} - X) \\
\leq \lambda R_n(\hat{X} - X) \leq \lambda R_d(\hat{X} - X)
\]

The first equality is from the duality inequality. The second inequality is because \( R_n(\Omega(D)) \leq \|\Omega(D)\|_2 \leq \|D\|_2 \).

Plugging the inequality back

\[
\|\Omega(\hat{X} - P_d(X^*))\|_2^2 \\
\leq \|\Omega(X - P_d(X^*))\|_2^2 + \lambda(R_d(X) - R_d(\hat{X})) + \lambda R_d(\hat{X} - X)) \\
\leq \|\Omega(X - X^*)\|_2^2 + 2\lambda R_d(X)
\]

91
where the last inequality follows from convexity. Choosing \( X = P_d(X^*) \) gives the conclusion in the theorem.

Next we bound the probability of \( R_n(M) \leq \lambda/4 \). First given an \( n \)th order tensor, we can divide the set \( \{1, 2, \ldots, n\} \) into a set \( a \) and its complement set \( b \), and obtain a matrix representation of the tensor in \( R^\otimes m_a \times R^\otimes m_b \). We denote the unfolding matrix of tensor \( X \) as \( X^{(a,b)} \) and let \( r_a \) and \( r_b \) be the dimension of the two modes respectively. The function \( R_n(M) \) satisfies \( R_n(M) \leq \|M^{(a,b)}\|_\infty \) for any set \( a \). Therefore, \( Pr(R_n(M) \geq \lambda/4) \leq Pr(M^{(a,b)} \geq \lambda/4) \).

Our observation model assumes \( E[\epsilon_i] = 0 \). In addition, since \( y_i \) are usually bounded, there exists a constant \( R \) such that \( |\epsilon_i| \leq 2R \). In this case, the observation tensor \( \Omega_i \) becomes a matrix \( Z_i \in \mathbb{R}^{r_a} \times \mathbb{R}^{r_b} \), and \( M^{(a,b)} = \sum_i \epsilon_i Z_i \). For simplicity, we assume uniform sampling of \( Z_i \) as our selection model for observation. This approach can also be used for non-uniform sampling. The next lemma is a generalization of Bernstein inequalities for matrices (see lemma 1 in (Koltchinskii et al., 2011)) which only requires an upper bound of the noise.

**Lemma 5.3.2.** Assume \( \epsilon_i \) satisfying \( E[\epsilon_i] = 0 \) and \( |\epsilon_i| \leq 2R \) almost surely for \( i = \{1, 2, \ldots, N\} \), then for any \( t > 0 \)

\[
\mathbb{P}(R_n(M) > 4R \max \left\{ \sqrt{\frac{t + \log(r_a + r_b)}{\min(r_a, r_b) N}}, \frac{2(t + \log(r_a + r_b))}{N} \right\}) \leq e^{-t}
\]

(5.20)

For large enough \( N \), the first term in the maximization dominates, and then we have:

\[
\mathbb{P}(R_n(M) > \lambda/4) \leq (d_a + d_b) \exp\left(-\frac{\lambda^2 N \min(d_a, d_b)}{16R^2}\right)
\]

(5.21)

which exponentially decreases with \( \lambda^2 \).

The analysis in Theorem (5.3.1) only bounds the empirical loss over the observation
set. To bound the distance between estimate $\hat{X}$ and $P_d(X^*)$, we need to make additional assumptions over $X^*$ and $\Omega_i$. Negahban et al. (2012) introduced restricted strong convexity (RSC) in which the objective function is strongly convex in a restricted set. The paper showed that with high probability RSC is satisfied under mild conditions. The analysis also applies to tensors, leading from the fact that we can always unfold a tensor into a matrix. Assuming RSC in our problem, then there exists a constant $\nu$,

$$
\|\hat{X} - P_d(X^*)\|_2^2 \leq \frac{\nu \prod_i m_i}{N} \|\Omega(\hat{X} - P_d(X^*))\|_2^2 \\
\leq \frac{2\nu \lambda \prod_i m_i}{N} (R_d(P_d(X^*)) + \frac{\lambda}{32})
$$

where the right side of equation is a constant depending $\|X^*\|$. As a point of difference from previous results for tensor and matrix completion, we have an additional constraint on the regularization parameter $\lambda \geq \|\Delta\|_2$. This is because our solution is biased towards tensors with low order components. In the extreme case that the tensor only contains higher order component such that $P_d(X) = 0$, the regularization $\lambda$ will force the solution to be $\hat{X} = 0$.

### 5.4 Alternating optimization algorithm

In this section, we develop an efficient algorithm for solving optimization problems (5.16). We propose two heuristic methods for selecting subsets of modes to optimize, and an alternating optimization algorithm to find a local optimum of parameters from the subset.

Replacing tensor $X$ with its decomposition form and using coordinate representation of $\Omega_i = (w^i_1, w^i_2, ..., w^i_n)$, the objective function (5.16) is equivalent to

$$
\min_{\kappa, \beta} \sum_{i=1}^{N} \left( y_i - \sum_{k} \kappa^{k} \prod_{j}(\beta^{k}_{j}(w^i_j)) \right)^2 + \lambda \sum_{k} |\kappa^{k}| 
$$

(5.22)
where $\beta^k_j(w^i_j)$ is the value of $\beta^k_j$ at position $w^i_j$. Each rank-one tensor $\otimes_{i=1}^N \beta^i_j$ can be considered as a coordinate in space $\mathbb{R}^{\otimes_j m_j}$, and $\kappa^i$ is the corresponding coefficient.

Starting from $\kappa^i = 0$ for all $i$, we use a coordinate descent algorithm that selects one coordinate in each iteration and optimizes the objective function along the coordinate.

At the $k^{th}$ iteration, the residue of $y_i$ is defined as

$$y^k_i = y_i - \sum_{i=1}^k \kappa^i \prod_{j=1}^m \beta^i_j(x^i_j) \quad (5.23)$$

Given a set of modes $S_k$ with size at most $d$, the optimization problem for a rank-one component is

$$\min_{\kappa, \beta} \left( y^k_i - \frac{\kappa^k}{\sqrt{\prod_{j:j \notin S_k} m_j}} \prod_{j \in S_k} \beta^i_j(x^i_j) \right)^2 + \lambda |\kappa^k| \quad (5.24)$$

Notice that the regularization $\lambda |\kappa^k|$ creates soft thresholding. The residue will have a smaller 2-norm as more components are added. Eventually we will find $\kappa^k = 0$ to be the optimum solution. The algorithm can then stop at this step.

We employ an alternating optimization algorithm to solve the optimization problem. To make the update more efficient, for each iteration we optimize $\kappa^k$ and one $\beta^k_j$ jointly. Let $\alpha = \kappa^k \beta^k_j$. The problem is

$$\min_{\alpha} \left( y^k_i - \frac{\alpha(x^i_j)}{\sqrt{\prod_{j:j \notin S_k} m_j}} \prod_{j \in S_k,j \neq v} \beta^k_j(x^i_j) \right)^2 + \lambda (\gamma \|\alpha\|^2 + \frac{1}{\gamma}) \quad (5.25)$$

where the last term is a variational form of $L_1$ regularization. When fixing $\gamma$ and $\beta$, the optimization problem reduces to a linear regression which has closed form
solutions.

\[
\alpha_p = \frac{\sqrt{\prod_{j: j \notin S_k} m_j \left( \sum_{i, x_i^k = p} y_i^k \prod_{j \in S_k, j \neq v} \beta_j^k (x_i^k) \right)}}{\sum_{i, x_i^k = p} \prod_{j \in S_k, j \neq v} (\beta_j^k (x_i^k))^2 + \lambda \gamma \sqrt{\prod_{j: j \notin S_k} m_j}}
\]

(5.26)

Given \( \alpha \), we have \( \kappa^k = \|\alpha\|_2 \) and \( \beta_v^k = \alpha / \kappa^k \). The alternating optimization is monotonic and in general converges to a local optimum. When \( d = 2 \) and the tensor are fully observed, the optimum \( \beta \) corresponds to the eigenvector with maximum eigenvalue \( \sigma_{\text{max}} \) of the matrix made by averaging the irreverent modes from the tensor. The solution to \( \kappa \) is thresholding \( \kappa = \max(0, \sigma_{\text{max}} - \lambda) \).

The other part of the algorithm is selecting the set \( S_k \) of \( \beta_j \) to optimize. A brute force search method would iterate over all possible combinations of \( d \) modes, and find the combination that decreases the objective function the most. It may still be efficient if the total number of modes \( n \) is small. When \( n \) is large, we propose two efficient heuristic methods to find a set of size \( d \).

**Method one**: assuming the initial assignment of each \( \beta_j \) is \( \frac{e}{\sqrt{m_j}} \), for each \( j \) we estimate the decrease of loss function by optimizing \( \beta_j \) and \( \kappa \) together while fixing the rest of \( \beta \). The decrease of loss function is used as a score for each mode. It estimates the impact to the objective function if the mode is selected to be optimized. The scores can be computed in \( O(Nm_k) \) time from (5.26). Next, we sort the scores and return the top-\( d \) modes. The advantage of the method is its efficiency because it considers a specific assignment of \( \beta_j \) and evaluates each mode separately. However, it ignores the joint effect of multiple modes.

**Method two**: Let the resulting tensor after averaging mode \( i \) be \( X(i) \in R^{\otimes_j \neq i m_j} \). More specifically, let \( X(i_1, ..., i_2) \) be the entry of \( X \) at position \( i_1, ..., i_2 \), then \( X(1)(i_2, ..., i_n) \) is the average of \( X(i_1, i_2, ..., i_n) \) over different \( i_1 \)'s from the training data. Next we measure \( \|X - e \otimes X(i)\|_2^2 \) and use the value as a score for mode \( i \). The value calculates to what extent the data would change if we remove one mode. In the extreme case that \( \|X - e \otimes X(i)\|_2^2 = 0 \), the mode \( i \) has no information at all and is therefore safe
to remove from the data. As before, we sort the scores and return the top-k modes. The criteria of this method is more natural. However, to compute the average of tensor entries, we may need to use a hash function to avoid enumerating all possible configurations. In the worst case, the size of the hash table is as large as the training data.

One point to emphasize is that the regularization function is very flexible. We can assign different regularization parameters to different sets of modes. Let \( \Lambda_S = \{ \otimes_j \beta_j, \beta_j \notin S = \frac{e}{\sqrt{m_j}}, \beta_j \in S \neq \frac{e}{\sqrt{m_j}} \} \) be the set of rank-one tensors that have non-ones on modes in \( S \). Then our weight function is \( W(\otimes^n_j=1 \beta^i_j) = W_S \) if \( \otimes^n_j=1 \beta^i_j \in \Lambda_S \). Then the weighted version of \( R_d(X) \) is

\[
R_{d,W}(X) = \min_{\kappa, \beta} W(\otimes^n_j=1 \beta^i_j) \sum_i |\kappa^i| \\
\text{s.t. } X = \sum_i \kappa^i \otimes^n_j=1 \beta^i_j, \otimes^n_j=1 \beta^i_j \in \cup_{k=1}^d \Phi_k
\]

where \( W(\cdot) \) is a weight function such that \( W(\otimes^n_j=1 \beta^i_j) = W_S \) if \( \otimes^n_j=1 \beta^i_j \in \Lambda_S \). By adding weights to rank one components for different subsets \( S \), we can vary the importance of different components in the problem.

### 5.5 Experimental results

In this section, we evaluate the proposed method on a real word recommendation system dataset. The goal is to recommend users with new movies that do not have any previous ratings. The dataset we use is Hetrec 2011 (Cantador et al., 2011) which contains user ratings from the Movielens dataset and movie features from IMDB. The movie features include directors, countries, actors, genres and user tags. Each movie has one director, one country, multiple actors, multiple genres and tags. The actors are ranked according to their positions in IMDB database.

For the recommendation task, we use a 4\(^{th}\)-order tensor composed of user, director,
country, and leading-actor, with its entries being rating values. Since our task is
to predict new movies, we do not make use of the movie ID for training. As a
preprocessing step, we remove actors that appeared less than five times in the cast
group, countries with a number of movies less than ten and movies with no leading
actors. The resulting data contain 371,550 ratings, 2,113 users, 2,377 movies, 1,059
directors, 25 countries and 1,184 actors.

Training and testing data are separated according to movie-ids. Ratings of 80 % of
the movies are used for training, and the ratings of the rest of the 20% are used for
testing. The number of ratings for training and testing data is 291,727 and 79,823
respectively.

We compare our algorithms to linear regression, nonlinear regression with second
order features and PARAFAC tensor factorization. For the linear regression model,
the categorical data are transformed into binary vectors. We add an additional $L_2$
regularization term $\lambda_r \|\theta\|^2$ to avoid over-fitting. $\lambda_r = 5$ is selected through cross
validation. For nonlinear regression with second order monomials, we iterate through
all pairs of four modes and include six additional features with size (243,973, 16,857,
203,425, 1,211, 1,927, 863). The features are apparently over-complete. For this
experiment, following the setting in linear regression, we add an additional $L_2$ regu-
larization. The parameters are $\lambda_1 = 5$ for order one features and $\lambda_2 = 40$ for order
two features. PARAFAC is a commonly used tensor factorization approach. We used
an alternating optimization approach implemented by Bader et al. (2012). We also
tried to comparing to (Liu et al., 2009), which uses a summation of a trace norm of
unfolded matrices as regularization, that is $\sum_{i=1}^{n} \|X^{(i)}\|_*$. However the algorithm in
(Liu et al., 2009) does not work well with sparse tensors. For the scale of our problem,
a full tensor is infeasible.

For our method, different regularization parameters for different order components are
selected. For simplicity, a scaled regularization parameter $\lambda' = \lambda \sqrt{\prod_{j=1}^{4} m_j}$ is used.
The parameter for order one components is $\lambda_1 = 3 \times 10^4$. To make fair comparison
with second order nonlinear regression, we consider \( d = 2 \). Because order two tensors can be seen as a combination of second order monomials, tensor completion and nonlinear regression models share the same feature space. The main difference being that for tensors there is a particular way of assigning weights to monomials.

The first step of the algorithm is to select the first four components as an order-one component for each mode. Then we use the first method in the algorithm section which greedily selects two modes to update. The regularization parameter is set to \( \lambda_2 = 5 \times 10^4 \).

From Figure 5-1, we see that PARAFAC fails to capture more information than linear regression. The nonlinear regression model is able to capture useful second order interactions among modes. Its training error is much smaller than other comparison methods. However, because the feature space is too large, the algorithm is not able to search for a good combination of monomials which have the generalization power to test data. On the contrary, the tensor completion approach we proposed with \( d = 2 \) has a bigger training error but smaller test error. It also achieves relatively good performance with only a few components.

Next we explore how regularization and set size \( d \) affects the performance. Figure 2 shows the results for \( d = 2, 3, 4 \). From figure 5-2, we see that larger regularization is needed for larger \( d \). One interesting observation is that the best performance is achieved at \( d = 2 \) while \( d = 4 \) performs the worst. One possible explanation is that the alternating optimization algorithm for optimizing rank-one components performs worse as nonlinearity in the objective function increases. Another explanation is that some of the modes have little prediction power. In this situation, we need to tune different regularization parameters for different modes. For \( d = 2 \), the way we greedily select a subset of modes to optimize has the effect of feature selection, which removes features with little information.
Figure 5-1: Test RMSE compared to linear regression, nonlinear regression model with second order monomials and PARAFAC tensor factorization method. The tensor completion approach with \( d=2 \) consistently performs better than the three methods with improvement \( (10^{-2}) : 2.66 \pm 0.26, 1.77 \pm 0.33 \) and \( 3.55 \pm 0.86 \).

Figure 5-2: Test RMSE as a function of regularization for various \( d \)

## 5.6 Conclusion

We cast set recommendation as a tensor factorization problem. A novel regularization function \( R_d(X) \) is proposed which could constrain both the number of rank-one components and the number of non-uniform modes in each rank-one component. Especially, \( R_2(X) \) can be computed efficiently and is a convex relaxation to the previously used pairwise rank constraint.

We develop an efficient algorithm for solving \( R_d(X) \) regularized optimization problems. The algorithm uses greedy methods to select a subset of modes to optimize. It
then performs alternating minimization to obtain parameter values for modes in the selected subset. Empirical results show our algorithm works well in practical problems. Although the feature space is the same for nonlinear regression and the tensor completion approach, our algorithm finds more generalizable feature combinations with this new regularization.
Chapter 6

Extension to dependency parsing

Traditionally, parsing research has been focused on predicting syntactic relations such as head-modifiers (arc) given features. This results in a high-dimensional vector representation even in the case of first order parsers. As a result, a great deal of research has been dedicated to feature engineering (Lazaridou et al., 2013; Marton et al., 2010, 2011). We depart from this approach and leverage high-dimensional feature vectors by mapping them onto low dimensional representation automatically. The low-dimensional embeddings are tailored to the syntactic context of words, and thus can be thought of as a proxy to manually constructed POS tags. In addition, because we automatically selected a small number of dimensions useful for parsing, we can leverage a wide range of correlated features such as word vectors from auxiliary data.

6.1 Low-rank dependency parsing

Let $x$ be a sentence and $\mathcal{Y}(x)$ the set of possible dependency trees over the words in $x$. We assume that the score $S(x, y)$ of each candidate dependency tree $y \in \mathcal{Y}(x)$
decomposes into a sum of “local” scores for arcs. Specifically:

\[ S(x, y) = \sum_{h \rightarrow m \in y} s(h \rightarrow m) \quad \forall y \in \mathcal{Y}(x) \]

where \( h \rightarrow m \) is the head-modifier dependency arc in the tree \( y \). Each \( y \) is understood as a collection of arcs \( h \rightarrow m \) where \( h \) and \( m \) are index words in \( x \).\(^1\) For example, \( x(h) \) is the word corresponding to \( h \). We suppress the dependence on \( x \) whenever it is clear from context. For example, \( s(h \rightarrow m) \) can depend on \( x \) in complicated ways as discussed below. The predicted parse is obtained as \( \hat{y} = \arg \max_{y \in \mathcal{Y}(x)} S(x, y) \).

The rank-1 tensor feature vector is constructed by taking the Kronecker product of simpler feature vectors associated with the head (vector \( \phi_h \in \mathbb{R}^n \)), and modifier (vector \( \phi_m \in \mathbb{R}^n \)), as well as the arc itself (vector \( \phi_{h,m} \in \mathbb{R}^d \), indicators for binned arc lengths).

\[ \phi_h \otimes \phi_m \otimes \phi_{h,m} \in \mathbb{R}^{n \times n \times d} \quad (6.1) \]

The arc score \( s_{\text{tensor}}(h \rightarrow m) \) associated with the tensor representation is defined as

\[ s_{\text{tensor}}(h \rightarrow m) = \langle A, \phi_h \otimes \phi_m \otimes \phi_{h,m} \rangle \quad (6.2) \]

We assume the tensor \( A \) has rank \( r \) such that \( A = \sum_{i=1}^{r} U(i,: ) \otimes V(i,:) \otimes W(i,:) \).

The score is then

\[ \sum_{i=1}^{r} [U \phi_h]_i [V \phi_m]_i [W \phi_{h,m}]_i \quad (6.3) \]

In addition to the tensor feature, we also include traditional binary features \( \phi_{h \rightarrow m} \in \mathbb{R}^L \) from MST/Turbo parser McDonald et al. (2005). The final arc score is defined as

\(^1\)Note that in the case of high-order parsing, the sum \( S(x, y) \) may also include local scores for other syntactic structures, such as grandhead-head-modifier score \( s(g \rightarrow h \rightarrow m) \). See Martins et al. (2013) for a complete list of these structures.
Unigram features:

<table>
<thead>
<tr>
<th></th>
<th>form</th>
<th>form-p</th>
<th>form-n</th>
</tr>
</thead>
<tbody>
<tr>
<td>lemma</td>
<td>lemma-p</td>
<td>lemma-n</td>
<td></td>
</tr>
<tr>
<td>pos</td>
<td>pos-p</td>
<td>pos-n</td>
<td></td>
</tr>
<tr>
<td>morph</td>
<td>bias</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bigram features:

<table>
<thead>
<tr>
<th></th>
<th>pos-p, pos</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>pos, pos-n</td>
</tr>
<tr>
<td></td>
<td>pos, lemma</td>
</tr>
<tr>
<td></td>
<td>morph, lemma</td>
</tr>
</tbody>
</table>

Trigram features:

|       | pos-p, pos, pos-n |

Table 6.1: Word feature templates used by our model. pos, form, lemma and morph stand for the fine POS tag, word form, word lemma and the morphology feature (provided in CoNLL format file) of the current word. There is a bias term that is always active for any word. The suffixes -p and -n refer to the left and right of the current word respectively. For example, pos-p means the POS tag to the left of the current word in the sentence.

\[
(1 - \gamma)s_{\text{tensor}}(h \rightarrow m) + \gamma s_{\phi}(h \rightarrow m)
\]

\[
= (1 - \gamma) \sum_{i=1}^{r}[U\phi_{h}]_{i}[V\phi_{m}]_{i}[W\phi_{h,m}]_{i} + \gamma \langle \theta, \phi_{h \rightarrow m} \rangle \tag{6.4}
\]

where $\theta \in \mathbb{R}^L$, $U \in \mathbb{R}^{r \times n}$, $V \in \mathbb{R}^{r \times n}$, and $W \in \mathbb{R}^{r \times d}$ are the model parameters to be learned. The rank $r$ and $\gamma \in [0,1]$ (balancing the two scores) represent hyper-parameters in our model.

### 6.2 learning

The training set $D = \{(\hat{x}_i, \hat{y}_i)\}_{i=1}^{N}$ consists of $N$ pairs, where each pair consists of a sentence $x_i$ and the corresponding target parse $y_i$. We adopt a maximum soft-margin framework for this learning problem. Specifically, we find parameters $\theta$, $U$, $V$, $W$, 

103
and \( \{\xi_i\} \) that minimize

\[
C \sum_i \xi_i + \|\theta\|^2 + \|U\|^2 + \|V\|^2 + \|W\|^2
\]

s.t. \( S_\gamma(\hat{x}_i, \hat{y}_i) \geq S_\gamma(\hat{x}_i, y_i) + \|\hat{y}_i - y_i\|_1 - \xi_i \)

\[\forall y_i \in \mathcal{Y}(\hat{x}_i), \forall i. \quad (6.5)\]

where \( \|\hat{y}_i - y_i\|_1 \) is the number of mismatched arcs between the two trees, and \( \xi_i \) is a non-negative slack variable. The constraints serve to separate the gold tree from other alternatives in \( \mathcal{Y}(\hat{x}_i) \) with a margin that increases with distance.

**Online Learning**  By applying an online learning setup, we update parameters successively sentence by sentence. In order to apply the passive-aggressive algorithm, we fix two of \( U, V \) and \( W \) (say, for example, \( V \) and \( W \)) in an alternating manner, and apply a closed-form update to the remaining parameters (here \( U \) and \( \theta \)). This is possible since the objective function with respect to \( (\theta, U) \) has a similar form as in the original passive-aggressive algorithm. To illustrate this, consider a training sentence \( x_i \). The update involves first finding the best competing tree,

\[
\tilde{y}_i = \arg \max_{y_i \in \mathcal{Y}(\hat{x}_i)} S_\gamma(\hat{x}_i, y_i) + \|\hat{y}_i - y_i\|_1
\]

which is the tree that violates the constraint in Eq. 6.5 most (i.e. maximizes the loss \( \xi_i \)). We then obtain parameter increments \( \Delta \theta \) and \( \Delta U \) by solving

\[
\min_{\Delta \theta, \Delta U, \xi \geq 0} \frac{1}{2} \|\Delta \theta\|^2 + \frac{1}{2} \|\Delta U\|^2 + C\xi
\]

s.t. \( S_\gamma(\hat{x}_i, \hat{y}_i) \geq S_\gamma(\hat{x}_i, \tilde{y}_i) + \|\hat{y}_i - \tilde{y}_i\|_1 - \xi \)

In this way, the optimization problem attempts to minimize the parameter change, while forcing it to achieve close to zero loss on this single instance. This problem has a closed form solution
\[ \Delta \theta = \min \left\{ C, \frac{\text{loss}}{\gamma^2 \|d\theta\|^2 + (1 - \gamma)^2 \|d\gamma\|^2} \right\} \gamma d\theta \]
\[ \Delta U = \min \left\{ C, \frac{\text{loss}}{\gamma^2 \|d\theta\|^2 + (1 - \gamma)^2 \|d\gamma\|^2} \right\} (1 - \gamma) d\gamma \]

where

\[ \text{loss} = S_\gamma(\hat{x}_i, \tilde{y}_i) + \|y_i - \tilde{y}_i\|_1 - S_\gamma(\hat{x}_i, \tilde{y}_i) \quad (6.7) \]
\[ d\theta = \sum_{h \rightarrow m \in \tilde{y}_i} \phi_{h \rightarrow m} - \sum_{h \rightarrow m \in \tilde{y}_i} \phi_{h \rightarrow m} \quad (6.8) \]
\[ d\gamma = \sum_{h \rightarrow m \in \tilde{y}_i} [(V \phi_m) \odot (W \phi_{h,m})] \odot \phi_h \quad (6.9) \]
\[ - \sum_{h \rightarrow m \in \tilde{y}_i} [(V \phi_m) \odot (W \phi_{h,m})] \odot \phi_h \quad (6.10) \]

where \((u \odot v)_i = u_i v_i\) is the Hadamard (element-wise) product. The magnitude of change of \(\theta\) and \(U\) is controlled by the parameter \(C\). By varying \(C\), we can determine an appropriate step size for the online updates. The updates also illustrate how \(\gamma\) balances the effect of the MST component of the score relative to the low-rank tensor score. When \(\gamma = 0\), the arc scores are entirely based on the low-rank tensor and \(\Delta \theta = 0\). Note that \(\phi_h, \phi_m, \phi_{h,m}\), and \(\phi_{h \rightarrow m}\) are typically very sparse for each word or arc. Therefore \(d\gamma\) and \(d\theta\) are also sparse and can be computed efficiently.

### 6.3 Experimental Setup

**Datasets** We test our dependency model on 14 languages, including the English dataset from CoNLL 2008 shared tasks and all 13 datasets from CoNLL 2006 shared tasks Buchholz and Marsi (2006); Surdeanu et al. (2008). These datasets include manually annotated dependency trees, POS tags and morphological information. Following standard practices, we encode this information as features.
<table>
<thead>
<tr>
<th></th>
<th>First-order only</th>
<th></th>
<th>High-order</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Ours</td>
<td>NT-1st</td>
<td>MST</td>
</tr>
<tr>
<td>Arabic</td>
<td>79.60</td>
<td>78.71</td>
<td>78.3</td>
</tr>
<tr>
<td>Bulgarian</td>
<td>92.30</td>
<td>91.14</td>
<td>90.98</td>
</tr>
<tr>
<td>Chinese</td>
<td>91.43</td>
<td>90.85</td>
<td>90.40</td>
</tr>
<tr>
<td>Czech</td>
<td>87.90</td>
<td>86.62</td>
<td>86.18</td>
</tr>
<tr>
<td>Danish</td>
<td>90.64</td>
<td>89.80</td>
<td>89.84</td>
</tr>
<tr>
<td>Dutch</td>
<td>84.81</td>
<td>83.77</td>
<td>82.89</td>
</tr>
<tr>
<td>English</td>
<td>91.84</td>
<td>91.40</td>
<td>90.59</td>
</tr>
<tr>
<td>German</td>
<td>90.24</td>
<td>89.70</td>
<td>89.54</td>
</tr>
<tr>
<td>Japanese</td>
<td>93.74</td>
<td>93.36</td>
<td>93.38</td>
</tr>
<tr>
<td>Portuguese</td>
<td>90.94</td>
<td>90.67</td>
<td>89.92</td>
</tr>
<tr>
<td>Slovene</td>
<td>84.25</td>
<td>83.15</td>
<td>82.09</td>
</tr>
<tr>
<td>Spanish</td>
<td>85.27</td>
<td>84.95</td>
<td>83.79</td>
</tr>
<tr>
<td>Swedish</td>
<td>89.86</td>
<td>89.66</td>
<td>88.27</td>
</tr>
<tr>
<td>Turkish</td>
<td>75.84</td>
<td>74.89</td>
<td>74.81</td>
</tr>
<tr>
<td>Average</td>
<td>87.76</td>
<td>87.05</td>
<td>86.5</td>
</tr>
</tbody>
</table>

Table 6.2: First-order parsing (left) and high-order parsing (right) results on CoNLL-2006 datasets and the English dataset of CoNLL-2008. For our model, the experiments are ran with rank $r = 50$ and hyper-parameter $\gamma = 0.3$. To remove the tensor in our model, we ran experiments with $\gamma = 1$, corresponding to columns NT-1st and NT-3rd. The last column shows results of most accurate parsers among (Nivre et al., 2006; McDonald et al., 2006; Martins et al., 2010, 2011, 2013; Koo et al., 2010; Rush and Petrov, 2012; Zhang and McDonald, 2012; Zhang et al., 2013).

**Methods** We compare our model to MST and Turbo parsers on non-projective dependency parsing. For our parser, we train both a first-order parsing model (as described in Section 3 and 4) as well as a third-order model. The third order parser simply adds high-order features, those typically used in MST and Turbo parsers, into our $s_\theta(x, y) = \langle \theta, \phi(x, y) \rangle$ scoring component. The decoding algorithm for the third-order parsing is based on Zhang et al. (2014). For the Turbo parser, we directly compare with the recent published results in Martins et al. (2013). For the MST parser, we train and test using the most recent version of the code.\(^2\) In addition, we implemented two additional baselines, NT-1st (first order) and NT-3rd (third order), corresponding to our model without the tensor component.

**Features** For the arc feature vector $\phi_{h \rightarrow m}$, the same set of feature templates as MST v0.5.1 are used. For head/modifier vector $\phi_h$ and $\phi_m$, we show the complete set of feature templates used by our model in Table 6.1. Finally, we use a similar set of

\(^2\)http://sourceforge.net/projects/mstparser/
feature templates as Turbo v2.1 for 3rd order parsing.

To add auxiliary word vector representations, we use the publicly available word vectors Cirik and Şensoy (2013), learned from raw data Globerson et al. (2007); Maron et al. (2010). Three languages in our dataset – English, German and Swedish – have corresponding word vectors in this collection. The dimensionality of this representation varies by language: English has 50 dimensional word vectors, while German and Swedish have 25 dimensional word vectors. Each entry of the word vector is added as a feature value into feature vectors $\phi_h$ and $\phi_m$. For each word in the sentence, we add its own word vector as well as the vectors of its left and right words.

We should note that because our model parameter $A$ is represented and learned in the low-rank form, we only have to store and maintain the low-rank projections $U\phi_h$, $V\phi_m$ and $W\phi_{h,m}$ rather than explicitly calculating the feature tensor $\phi_h \otimes \phi_m \otimes \phi_{h,m}$. Therefore updating parameters and decoding a sentence is still efficient, i.e., linear in the number of values of the feature vector. In contrast, if we take the cross-product of the auxiliary word vector values, POS tags and lexical items of a word and its context, and then add the crossed values into a normal model (in $\phi_{h \rightarrow m}$), the number of features for each arc would be at least quadratic, growing into thousands, and would be a significant impediment to parsing efficiency.

**Evaluation** Following standard practices, we train our full model and the baselines for 10 epochs. As the evaluation measure, we use unlabeled attachment scores (UAS) excluding punctuation. In all the reported experiments, the hyper-parameters are set as follows: $r = 50$ (rank of the tensor), $C = 1$ for the first-order model and $C = 0.01$ for the third-order model.
6.4 Results

**Overall Performance** Table 6.2 shows the performance of our model and the baselines on 14 CoNLL datasets. Our model outperforms Turbo parser, MST parser, as well as its own variants without the tensor component. The improvements of our low-rank model are consistent across languages, results for the first order parser are better on 11 out of 14 languages. By comparing NT-1st and NT-3rd (models without low-rank) with our full model (with low-rank), we obtain 0.7% absolute improvement on first-order parsing, and 0.3% improvement on third-order parsing. Our model also achieves the best UAS on 5 languages.

We next focus on the first-order model and gauge the impact of the tensor component. First, we test our model by varying the hyper-parameter $\gamma$ which balances the tensor score and the traditional MST/Turbo score components. Figure 6-1 shows the average UAS on CoNLL test datasets after each training epoch. We can see that the improvement of adding the low-rank tensor is consistent across various choices of hyper parameter $\gamma$. When training with the tensor component alone ($\gamma = 0$), the model converges more slowly. Learning the tensor is more difficult because the scoring function is not linear (nor convex) with respect to parameters $U$, $V$ and $W$. However, the tensor scoring component achieves better generalization on the test data, resulting in better UAS than NT-1st after 8 training epochs.

To assess the ability of our model to incorporate a range of features, we add unsupervised word vectors to our model. As described in the previous section, we do so by appending the values of different coordinates in the word vector into $\phi_h$ and $\phi_m$. As Table 6.3 shows, adding this information increases the parsing performance for all of the three languages. For instance, we obtain more than 0.5% absolute improvement on Swedish.

**Syntactic Abstraction without POS** Because our model learns a compressed representation of feature vectors, we are interested in measuring its performance
Figure 6-1: Average UAS on CoNLL testsets after different epochs. Our full model consistently performs better than NT-1st (its variation without tensor component) under different choices of the hyper-parameter $\gamma$.

<table>
<thead>
<tr>
<th></th>
<th>no word vector</th>
<th>with word vector</th>
</tr>
</thead>
<tbody>
<tr>
<td>English</td>
<td>91.84</td>
<td>92.07</td>
</tr>
<tr>
<td>German</td>
<td>90.24</td>
<td>90.48</td>
</tr>
<tr>
<td>Swedish</td>
<td>89.86</td>
<td>90.38</td>
</tr>
</tbody>
</table>

Table 6.3: Results of adding unsupervised word vectors to the tensor. Adding this information yields consistent improvement for all languages.

<table>
<thead>
<tr>
<th></th>
<th>Our model</th>
<th>NT-1st</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>-POS + wv.</td>
<td>-POS  +POS</td>
</tr>
<tr>
<td>English</td>
<td>88.89</td>
<td>90.49</td>
</tr>
<tr>
<td>German</td>
<td>82.63</td>
<td>85.80</td>
</tr>
<tr>
<td>Swedish</td>
<td>81.84</td>
<td>85.90</td>
</tr>
</tbody>
</table>

Table 6.4: The first three columns show parsing results when models are trained without POS tags. The last column gives the upper-bound, i.e. the performance of a parser trained with 12 Core POS tags. The low-rank model outperforms NT-1st by a large margin. Adding word vector features further improves performance.

when part-of-speech tags are not provided (See Table 6.4). The rationale is that given all other features, the model would induce representations that play a similar role to POS tags. Note that the performance of traditional parsers drops when tags are not provided. For example, the performance gap is 10% on German. Our experiments show that low-rank parser operates effectively in the absence of tags. In fact, it nearly reaches the performance of the original parser which had used the tags on English.
<table>
<thead>
<tr>
<th>greatly</th>
<th>profit</th>
<th>says</th>
<th>on</th>
<th>when</th>
</tr>
</thead>
<tbody>
<tr>
<td>actively</td>
<td>earnings</td>
<td>adds</td>
<td>with</td>
<td>where</td>
</tr>
<tr>
<td>openly</td>
<td>franchisees</td>
<td>predicts</td>
<td>into</td>
<td>what</td>
</tr>
<tr>
<td>significantly</td>
<td>shares</td>
<td>noted</td>
<td>at</td>
<td>why</td>
</tr>
<tr>
<td>outright</td>
<td>revenue</td>
<td>wrote</td>
<td>during</td>
<td>which</td>
</tr>
<tr>
<td>substantially</td>
<td>members</td>
<td>contends</td>
<td>over</td>
<td>who</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>increase</th>
<th>will increase again</th>
<th>an increase of</th>
</tr>
</thead>
<tbody>
<tr>
<td>rise</td>
<td>arguing</td>
<td>gain</td>
</tr>
<tr>
<td>advance</td>
<td>be</td>
<td>prices</td>
</tr>
<tr>
<td>contest</td>
<td>charging</td>
<td>payment</td>
</tr>
<tr>
<td>halt</td>
<td>gone</td>
<td>members</td>
</tr>
<tr>
<td>Exchequer</td>
<td>making</td>
<td>subsidiary</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>hit</th>
<th>attacks hit the</th>
<th>hardest hit is</th>
</tr>
</thead>
<tbody>
<tr>
<td>shed</td>
<td>distributes</td>
<td>monopolies</td>
</tr>
<tr>
<td>rallied</td>
<td>stayed</td>
<td>pills</td>
</tr>
<tr>
<td>triggered</td>
<td>sang</td>
<td>sophistication</td>
</tr>
<tr>
<td>appeared</td>
<td>removed</td>
<td>ventures</td>
</tr>
<tr>
<td>understate</td>
<td>eased</td>
<td>factors</td>
</tr>
</tbody>
</table>

Table 6.5: Five closest neighbors of the queried words (shown in bold). The upper part shows our learned embeddings group words with similar syntactic behavior. The two bottom parts of the table demonstrate how the projections change depending on the syntactic context of the word.

**Examples of Derived Projections**  We now manually analyze low-dimensional projections to assess whether they capture syntactic abstraction. For this purpose, we train a model with only a tensor component, such that it has to learn an accurate tensor, on the English dataset and obtain low dimensional embeddings $U\phi_w$ and $V\phi_w$ for each word. The two r-dimension vectors are concatenated as an “averaged” vector. We use this vector to calculate the cosine similarity between words. Table 6.5 shows examples of the five closest neighbors of queried words. While these lists include some noise, we can clearly see that the neighbors exhibit similar syntactic behavior. For example, “on” is close to other prepositions. More interestingly, we can consider the impact of syntactic context on the derived projections. The bottom part of Table 6.5 shows that the neighbors change substantially depending on the syntactic role of the word. For example, the closest words to the word “increase” are verbs in the contextualized phrase, “will increase again”, while the closest words become nouns given a different phrase such as,“an increase of”.

110
Table 6.6: Comparison of training times across three typical datasets. The second column is the number of tokens in each data set. The third column shows the average sentence length. Both first-order models are implemented in Java and run as a single process.

<table>
<thead>
<tr>
<th></th>
<th>#Tok.</th>
<th>Len.</th>
<th>Train. Time (hour)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arabic</td>
<td>42K</td>
<td>32</td>
<td>0.13</td>
</tr>
<tr>
<td>Chinese</td>
<td>337K</td>
<td>6</td>
<td>0.37</td>
</tr>
<tr>
<td>English</td>
<td>958K</td>
<td>24</td>
<td>1.88</td>
</tr>
</tbody>
</table>

Table 6.6 illustrates the impact of estimating low-rank tensor parameters on the running time of the algorithm. For comparison, we also show the NT-1st times across three typical languages. The Arabic dataset has the longest average sentence length, while the Chinese dataset has the shortest sentence length in CoNLL 2006. Based on these results, estimating a rank-50 tensor together with MST parameters only increases the running time by a factor of 1.7.
Appendix A

Supplementary materials

Proof of lemma 3.3.1

Proof. In this case, $F(B)$ is differentiable with gradient

$$dF(B) = 1/2(\lambda^2 I - Q^TQ)$$

where $Q = Q(B)$ (unique) is obtained from solving the maximization problem. The condition $|d(Loss(z))| \to \infty$ only if $|z| \to \infty$ implies that when $|q| \to \infty$, its corresponding primal solution $|z(q)| = |d(Loss^*(q))| \to \infty$. Since $(qz - Loss^*(q))$ is also upper bounded by $Loss(z)$, we have $(qz - Loss^*(q)) \to -\infty$ as $q \to \infty$. Therefore the unique maximizer $Q(B)$ of (3.19) remains bounded, i.e. $\|Q(B)\| \leq D$ for any $B$. Moreover, since $\text{tr}((\lambda^2 I - Q^TQ)B) > 0$, the difference between derivative can be bounded as

$$\|dF(B_1) - dF(B_2)\| = 1/2\|(Q^1)^TQ^1 - (Q^2)^TQ^2\| \leq D\|Q^1 - Q^2\|$$

Each row of $Q$ can be learned independently by solving

$$Y_{u,I_u} - d(Loss^*(Q_{u,I_u})) - Q_{u,I_u}B = 0$$
The equations are satisfied for all \( Q(B) \). Therefore for any \( B_1 \) and \( B_2 \), we have

\[
d(\text{Loss}^*(Q^{1}_{u,I_u})) + Q^{1}_{u,I_u}B_1 = d(\text{Loss}^*(Q^{2}_{u,I_u})) + Q^{2}_{u,I_u}B_2 \tag{A.4}
\]

From the mean value theorem, there exist \( \tilde{Q} \) and positive semi-definite matrix \( \tilde{B} \), such that

\[
Q^{1}_{u,I_u}B_1 - Q^{2}_{u,I_u}B_2 = (Q^{1}_{u,I_u} - Q^{2}_{u,I_u})\tilde{B} + \tilde{Q}_{u,I_u}(B^1 - B^2). \tag{A.4}
\]

From the duality, we get \( q = d(\text{Loss}(z(q))) \) and \( d(\text{Loss}^*(q)) = z(q) \) where \( z(q) \) is the solution to the maximization problem. Because the loss function has Lipschitz continuous derivative, there exists constant \( c \) such that

\[
|q^1 - q^2| = |d(\text{Loss}(z(q^1))) - d(\text{Loss}(z(q^2)))| \leq c|z(q^1) - z(q^2)|
\]

\[
= c|d(\text{Loss}^*(q^1)) - d(\text{Loss}^*(q^2))| \tag{A.5}
\]

Let \( G \) be a diagonal matrix with

\[
G_{i,i} = \frac{d(\text{Loss}^*(Q^{1}_{u,I_u(i)})) - d(\text{Loss}^*(Q^{2}_{u,I_u(i)}))}{Q^{1}_{u,I_u(i)} - Q^{2}_{u,I_u(i)}} \tag{A.6}
\]

From the convexity of \( \text{Loss}^* \) function and above results, we have \( 0 \leq G_{i,i} \leq 1/c \). Taking it back to (A.4) yields,

\[
Q^{1}_{u,I_u} - Q^{2}_{u,I_u} = \tilde{Q}_{u,I_u}(B^1 - B^2)(G + \tilde{B})^{-1} \tag{A.7}
\]

\( \tilde{Q}_{u,I_u} \) is upper bounded by \( D \) and \( (G + \tilde{B})^{-1} \leq cI \), therefore \( \|Q^{1}_{u,I_u} - Q^{2}_{u,I_u}\| \) is upper bounded by \( \|B^1 - B^2\| \) with constant difference, so is \( \|Q^1 - Q^2\| \). Combined with (A.2), \( F(B) \) has a Lipschitz continuous derivative.

**Proof of theorem 3.3.2**

*Proof.* Consider \( (B^0, Q^0), (B^1, Q^1), \ldots \) the sequence generated by the primal-dual algorithm. Since \( dF(B) \) is Lipschitz continuous with some constant \( L \), \( F(B) \) has a
quadratic upper bound:

\[ F(B) \leq F(B^{r-1}) + \langle dF(B^{r-1}), B - B^{r-1} \rangle + L/2\|B - B^{r-1}\|_F^2 \]  

(A.8)

In each iteration, we add a constraint \((b^r)^TQ^TQb^r \leq \lambda^2 \) and fully optimize \(Q\) and \(\xi_i\) to satisfy all the constraints that have been added. Prior to including the \(r^{th}\) constraint, from complementary slackness, we have for each \(i < r\), \(\xi_i^i(b^r)^T(\lambda^2 I - Q^TQ)b^i = 0\) where \(Q\) is optimized based on \(r - 1\) constraints. This means that \(tr((\lambda^2 I - Q^TQ)\sum_{i=0}^{r-1}\xi_i^i(b^r)^T) = tr((\lambda^2 I - Q^TQ)B^{r-1}) = 0, \) or \(\langle dF(B^{r-1}), B^{r-1} \rangle = 0\).

Let \(B^r = B^{r-1} + \xi^r b^r(b^r)^T\), where \(b^r = \arg\max_{\|b\|=1} b^TQ^TQb\), i.e., the largest eigenvalue \(\lambda_r^2\ (\lambda_r > \lambda)\). Here, \(\lambda_r\) is the largest eigenvalue prior to including \(b^r\). As a result, the upper bound evaluated at \(B^r\) yields

\[ F(B^r) \leq F(B^{r-1}) + \xi^r(\lambda^2 - \lambda_r^2) + 2L(\xi^r)^2 \]

By setting \(\xi^r = (\lambda_r^2 - \lambda^2)/4L\) (non-negative), we get

\[ F(B^r) \leq F(B^{r-1}) - (\lambda_r^2 - \lambda^2)^2/8L \]

The optimal value of \(\xi^r\) may be different. Moreover, at each iteration, \(B^r\), including all of its previous constraints, are optimized. Thus the actual decrease may be somewhat larger.

From the convexity of \(F(B)\) and complementary slackness, we have

\[ F(B^{r-1}) - F(B^*) \leq \langle dF(B^{r-1}), B^{r-1} - B^* \rangle = -\langle dF(B^{r-1}), B^* \rangle \]
where $B^*$ is the optimum. $B^*$ is a positive semi-definite matrix. Therefore

$$-\langle dF(B_r^{-1}), B^* \rangle \leq -\langle \text{Proj}_{\text{neg}}(dF(B_r^{-1})), B^* \rangle \leq (\lambda_r^2 - \lambda^2) \text{tr}(B^*) = (\lambda_r^2 - \lambda^2)C$$

where $\text{Proj}_{\text{neg}}(\cdot)$ is a projection to negative semi-definite matrices. The minimum eigenvalue of $dF(B_r^{-1}) = 1/2(\lambda^2 - Q^TQ)$ is $\lambda^2 - \lambda_r^2$. Combining this with the sufficient decrease, we get

$$F(B_r^{-1}) - F(B^r) \geq \frac{(F(B_r^{-1}) - F(B^*))^2}{8LC^2}$$

The above inequality implies

$$\frac{1}{F(B^r) - F(B^*)} - \frac{1}{F(B_r^{-1}) - F(B^*)} \geq \frac{F(B_r^{-1}) - F(B^*)}{8LC^2(F(B^r) - F(B^*))} \geq \frac{1}{8LC^2}$$

Summing over all $r$, we get

$$F(B^r) - F(B^*) \leq \frac{8LC^2(F(B^0) - F(B^*))}{r(F(B^0) - F(B^*))} + 8LC^2 = O\left(\frac{1}{r}\right)$$

**Proof of Theorem 4.2.2**

**Proof.** From the optimality of $\hat{X}$,

$$\|Y - \hat{X}\|_\Omega^2 + \frac{\lambda}{\sqrt{mn}} \|\hat{X}\|_* \leq \|Y - \hat{X}\|_\Omega^2 + \frac{\lambda}{\sqrt{mn}} \|\hat{X}\|_*$$  \hspace{1cm} (A.9)

Plug in $Y = \hat{X} + \epsilon$ and reorganize the terms,

$$\|\Delta\|_\Omega^2 - \frac{1}{N} \sum_{(i,j) \in \Omega} \epsilon_{i,j} \Delta_{i,j} \leq \frac{\lambda}{\sqrt{mn}} (\|\hat{X}\|_* - \|\hat{X} + \Delta\|_*)$$  \hspace{1cm} (A.10)
From lemma 4.2.1, we have

$$\frac{1}{N} \sum_{(i,j) \in \Omega} \epsilon_{i,j} \Delta_{i,j} \leq \frac{1}{\sqrt{N}} \|\epsilon\|_\infty \|\Delta\|_{\Omega} \leq 2\sigma \sqrt{\frac{\log N}{N}} \|\Delta\|_{\Omega}$$  \hspace{1cm} (A.11)

Substituting in (A.10),

$$(\|\Delta\|_{\Omega} - c\sigma \sqrt{\frac{\log N}{N}})^2 \leq \frac{\lambda}{\sqrt{mn}} (\|\hat{X}\|_* - \|\hat{X} + \Delta\|_* ) + \frac{c^2\sigma^2 \log N}{N}$$

$$\leq \frac{\lambda}{\sqrt{mn}} \|\Delta\|_* + \frac{c^2\sigma^2 \log N}{N}$$  \hspace{1cm} (A.12)

where we have used triangle inequality for trace norm. It then follows that

$$\|\Delta\|_{\Omega}^2 \leq \left( \sqrt{\frac{\lambda}{\sqrt{mn}} \|\Delta\|_* + \frac{c^2\sigma^2 \log N}{N}} + c\sigma \sqrt{\frac{\log N}{N}} \right)^2 \leq 2 \frac{\lambda}{\sqrt{mn}} \|\Delta\|_* + \frac{4c^2\sigma^2 \log N}{N}$$  \hspace{1cm} (A.13)

By the RSC assumption,

$$\|\Delta\|_{(F)}^2 \leq \frac{2\lambda}{\kappa \sqrt{mn}} \|\Delta\|_* + \frac{4c^2\sigma^2 \log N}{\kappa N}$$  \hspace{1cm} (A.14)

From lemma 4.2.1, and consider $\pi_B(\Delta)$ has rank at most $2r$,

$$\|\Delta\|_* \leq \|\pi_B(\Delta)\|_* + \|\pi_B(\Delta)\|_* \leq 2\|\pi_B(\Delta)\|_* + \frac{c^2\sigma^2 \sqrt{mn}}{N\lambda} \log N$$

$$\leq 2\sqrt{2rmn} \|\Delta\|_{(F)} + \frac{c^2\sigma^2 \sqrt{mn}}{N\lambda} \log N$$  \hspace{1cm} (A.15)

Substituting the bound in (A.14),

$$\|\Delta\|_{(F)} \leq c\sigma \sqrt{\frac{6\log N}{\kappa N}} + \frac{4\sqrt{2r}}{\kappa} \lambda$$  \hspace{1cm} (A.16)

□
Proof of Theorem 4.2.5

**Proof.** We first consider the singular values in \((\lambda I + \Sigma)^{-1}\). Let \(Q = [\hat{Q}, \hat{Q}^\perp]\) and \(\Delta = \Sigma - \hat{\Sigma}\),

\[
Q^T(\lambda I + \Sigma)^{-1}Q = \begin{pmatrix} \lambda I + \hat{\Sigma} + \Delta_{1,1} & \Delta_{1,2} \\ \Delta_{2,1} & \lambda I + \Delta_{2,2} \end{pmatrix}^{-1} = \begin{pmatrix} B_1 & B_2 \\ B_2^T & B_3 \end{pmatrix} \tag{A.17}
\]

Extending the inversion yields

\[
B_1 = (\lambda I + \hat{\Sigma} + \Delta_{1,1} - \Delta_{1,2}(\lambda I + \Delta_{2,2})^{-1}\Delta_{2,1})^{-1}
\]

\[
B_3 = (\lambda I + \Delta_{2,2} - \Delta_{2,1}(\lambda I + \hat{\Sigma} + \Delta_{1,1})^{-1}\Delta_{1,2})^{-1} \tag{A.18}
\]

Due to symmetry \(\Delta_{1,2} = \Delta_{2,1}^T\). \(\Sigma\) is positive semi-definite, \(\lambda I + \Delta_{2,2} \succeq \lambda I\), then \(\Delta_{1,2}(\lambda I + \Delta_{2,2})^{-1}\Delta_{2,1} \preceq \Delta_{1,2}\Delta_{2,1}^T/\lambda\). Combining with \(\delta^2 \geq \|\Delta\|_F^2 \geq \|\Delta_{1,1}\|_F^2 + \|\Delta_{1,2}\|_F^2\), we can lower bound the minimum eigenvalue of \(B_1^{-1}\) and obtain

\[
\|B_1\|_{op} \leq \frac{1}{\beta - \delta^2/\lambda} \tag{A.19}
\]

The semi-definite condition of \(\Sigma\) is important to get the bound, otherwise because \(\delta\) and \(\lambda\) are usually on the same scale which can make \(\|B_1\|_{op}\) unbounded. On the other hand, it’s easy to see \(B_3^{-1} \preceq \lambda I + \Delta_{2,2} \preceq (\lambda + \delta)I\), therefore the minimum eigenvalue of \(B_3\) is lower bounded by \(\frac{1}{\lambda + \delta}\). In sum, for any normalized vector \(v\), if \(v \in \hat{Q}, v^T(\lambda I + \Sigma)^{-1}v \leq \frac{1}{\beta - \delta^2/\lambda}v\), and if \(v \in \hat{Q}^\perp, v^T(\lambda I + \Sigma)^{-1}v \geq \frac{1}{\lambda + \delta}\).

For simplicity, we re-organize the index to make observation set \(k\) in the front. The prediction problem can be formulated as a minimization problem,

\[
\hat{x}_k = \arg\min_y [x_k, y](\lambda I + \Sigma)^{-1}[x_k, y]^T \tag{A.20}
\]

To see the equivalence, one can take the derivative to \(y\) and let it be 0. Given \(\hat{x}_k\) and
\( \hat{x}_k \), we have

\[
[x_k, \hat{x}_k^c](\lambda I + \Sigma)^{-1}[x_k, \hat{x}_k^c]^T \leq [x_k, \hat{x}_k^c](\lambda I + \Sigma)^{-1}[x_k, \hat{x}_k^c]^T \tag{A.21}
\]

from optimality. Use some linear algebra and let \( dx = \hat{x}_k^c - \hat{x}_k^c \),

\[
0 \geq [0, dx](\lambda I + \Sigma)^{-1}[0, dx]^T + 2[x_k, \hat{x}_k^c](\lambda I + \Sigma)^{-1}[0, dx]^T \\
\geq ||[0, dx](\lambda I + \Sigma)^{-\frac{1}{2}}||_F^2 - 2||[x_k, \hat{x}_k^c](\lambda I + \Sigma)^{-\frac{1}{2}}||_F ||[0, dx](\lambda I + \Sigma)^{-\frac{1}{2}}||_F \tag{A.22}
\]

Since \( \hat{x} \in \hat{Q} \) and \( x_k = \hat{x}_k^c + \epsilon_k \), as we show before

\[
||x(\lambda I + \Sigma)^{-\frac{1}{2}}||_F \leq \sqrt{\frac{1}{\beta - \delta^2/\lambda}}||\hat{x}||_F + \frac{||\epsilon_k||_F}{\sqrt{\lambda'}} \tag{A.23}
\]

The second term is because the maximum eigenvalue of \((\lambda I + \Sigma)^{-1}\) is bounded by \(1/\lambda'\). Substituting it to (A.22),

\[
||[0, dx](\lambda I + \Sigma)^{-\frac{1}{2}}||_F \leq 2\left( \sqrt{\frac{1}{\beta - \delta^2/\lambda}}||\hat{x}||_F + \frac{||\epsilon_k||_F}{\sqrt{\lambda'}} \right) \tag{A.24}
\]

Assume \([0, dx] = [x_1, y_1] + [-x_1, y_2] \) where \([x_1, y_1] \in \hat{Q} \) and \([-x_1, y_2] \in \hat{Q}^\perp \). Plug it in,

\[
2\left( \sqrt{\frac{1}{\beta - \delta^2/\lambda}}||\hat{x}||_F + \frac{||\epsilon_k||_F}{\sqrt{\lambda'}} \right) \geq \|[x_1, y_2](\lambda I + \Sigma)^{-\frac{1}{2}}\|_F - \|[x_1, y_1](\lambda I + \Sigma)^{-\frac{1}{2}}\|_F \\
\geq \frac{1}{\lambda' + \delta}||[x_1, y_2]||_F - \frac{1}{\beta - \delta^2/\lambda' ||[x_1, y_1]||_F} \\
\geq \left( \sqrt{\frac{1}{\lambda' + \delta}} - \sqrt{\frac{\gamma |k|}{(\beta - \delta^2/\lambda')m}} \right)||x_1||_F \tag{A.25}
\]

where the second inequality comes from our assumption that \( x \) is in the feasible prediction set \( C(\gamma) \) and our previous analysis of eigenvalues of \((\lambda I + \Sigma)^{-1}\). Then we
can bound the prediction difference as

\[
\| d_x \|_F \leq \| x_1, y_1 \|_F + \| - x_1, y_2 \|_F \\
\leq \| x_1, y_1 \|_F + \sqrt{\frac{\lambda' + \delta}{\beta - \delta^2/\lambda'}} (2\| \hat{x} \|_F + \| [x_1, y_1] \|_F) + \sqrt{\frac{\lambda' + \delta}{\lambda}} \| \epsilon_k \|_F \\
\leq \sqrt{\frac{\gamma|k|}{m}} (1 + \sqrt{\frac{\lambda' + \delta}{\beta - \delta^2/\lambda'}}) \| x_1 \|_F + 2\sqrt{\frac{\lambda' + \delta}{\beta - \delta^2/\lambda'}} \| \hat{x} \|_F + 2\sqrt{\frac{\lambda' + \delta}{\lambda'}} \| \epsilon_k \|_F
\]

Substituting in the inequality of \( \| x_1 \|_F \) will gives us a bound of \( \| dx \|_F \) as a function of \( \| \hat{x} \|_F \) and \( \| \epsilon_k \|_F \). \( \epsilon^2 \) follows sub-exponential distribution with norm \( \| \epsilon \| \leq 2\sigma^2 \).

Using a Bernstein-type inequality (see Proposition 5.16 in Vershynin (2010)), with probability at least \( 1 - \exp(-c_2^2) \)

\[
\| \epsilon_k \|_F^2 \leq c_2^2 \sigma^2 \sqrt{k}
\]

(A.26)

To simply the bound, assuming \( \beta \gg \delta^2/\lambda' \), then

\[
\| \hat{x}_{k^c} - \hat{x}_{k^c} \|_F = \| dx \|_F \leq 2\sqrt{\lambda' + \delta} (\sqrt{\frac{\gamma|k|}{m}} + 1) (\| \hat{x} \|_F \sqrt{\beta} + \frac{c_1 \sigma k^{1/4}}{\sqrt{\lambda'}})
\]

(A.27)
Bibliography


C. Ding, T. Li, and W. Peng. On the equivalence between non-negative matrix factorization and probabilistic latent semantic indexing. Computational Statistics and Data Analysis, 2008.


S. Shalev-Shwartz, A. Gonenand, and O. Shamir. Large-scale convex minimization with a low-rank constraint. In ICML, 2011.


