Blind Regression: Understanding Collaborative Filtering from Matrix Completion to Tensor Completion

by

Yihua Li

Submitted to the Department of Electrical Engineering and Computer Science on May 20, 2016 in Partial Fulfillment of the Requirements for the Degree of Master of Engineering in Electrical Engineering and Computer Science

Abstract

Neighborhood-based Collaborative filtering (CF) methods have proven to be successful in practice and are widely applied in commercial recommendation systems. Yet theoretical understanding of their performance is lacking. In this work, we introduce a new framework of Blind Regression which assumes that there are latent features associated with input variables, and we observe outputs of some Lipschitz continuous function over those unobserved features.

We apply our framework to the problem of matrix completion and give a nonparametric method which, similar to CF, combines the local estimates according to the distance between the neighbors. We use the sample variance of the difference in ratings between neighbors as the proximity of the distance. Through error analysis, we show that the minimum sample variance is a good proxy of the prediction error in the estimates. Experiments on real-world datasets suggests that our matrix completion algorithm outperforms classic user-user and item-item CF approaches.

Finally, our framework easily extends to the setting of higher-order tensors and we present our algorithm for tensor completion. The result from real-world application of image inpainting demonstrates that our method is competitive with the state-of-the-art tensor factorization approaches in terms of predictive performance.

Thesis Supervisor: Devavrat Shah
Title: Associate Professor in Electrical Engineering and Computer Science
Acknowledgements

First, I would like to thank my adviser, Devavrat Shah. His guidance over the past year has been truly invaluable and I am very lucky to have had the opportunity to work with him at MIT.

Among the group members, I would especially like to thank Christina Lee and Dogyoon Song. I am grateful for their insightful comments, suggestions and ideas on the problem we are working on, as well as the support along the way.

Lastly, I would like to thank my family and friends, in particular, my parents and my sister Yixin. They have provided me the encouragement and love to get through my graduate study. Thank you for always being there for me.
# Contents

Abstract iii

Acknowledgments iv

List of Figures ix

1 Introduction 1
   1.1 Problem .................................................. 1
   1.2 Contributions ........................................... 2
   1.3 Thesis Organization ..................................... 3

2 Background 5
   2.1 Matrix Completion ....................................... 5
      2.1.1 Collaborative Filtering ............................. 5
      2.1.2 Matrix Factorization ............................... 6
   2.2 Tensor Completion ....................................... 7
   2.3 Kernel Regression ....................................... 7

3 Setting 9
   3.1 Model and Assumptions ................................. 9
   3.2 Notations ................................................ 10
   3.3 Evaluation Metrics ..................................... 10

4 Algorithm 13
   4.1 Intuition ................................................ 13
   4.2 Matrix Completion Algorithm ........................... 14
   4.3 Extension to Higher Dimensions ....................... 17

5 Analysis 19
   5.1 Sample Complexity Analysis ........................... 19
   5.2 Error Analysis .......................................... 20
List of Figures

6.1 Performance of Algorithms on MovieLens Data Set. \( \lambda \) values used by our algorithm are 2.8 (10%), 2.3 (30%), 1.7 (50%), 1 (70%). .................................................. 25
6.2 Performance of Algorithms on Netflix Data Set. \( \lambda \) values used by our algorithm are 1.8 (10%), 1.7 (30%), 1.6 (50%), 1.5 (70%) for Netflix. .................................................. 25
6.3 Effect of \( \lambda \) on RMSE from MovieLens data set (10% evaluation set). .................................................. 26
6.4 Distribution of minimum sample variance (\( \beta = 5 \)). .................................................. 27
6.5 Variation of Squared Error across \( \min_v s_{uv}^2 \) buckets (\( \beta = 5 \)). The red line plots the median, while the box shows the 25\(^{th}\) and 75\(^{th}\) percentile, and end of dashed lines extends to the most extreme data point within 1.5 interquartile range of the upper quartile. .................................................. 27
6.6 Variance of Prediction Error vs. minimum sample variance (\( \beta = 5 \)) .................................. 28
6.7 Images used in Tensor Completion experiments. .......................................................... 29
6.8 Performance comparison between different approaches of our tensor completion algorithm based on RSE vs missing rate. .................................................. 29
6.9 Performance comparison between different tensor completion algorithms based on RSE vs test set size. .................................................. 30
6.10 Sample recovery results for Facade image. 50 %, 70 % and 90% of values are removed for prediction for each row respectively. The first image in each row shows the original image with missing values, while the rest is the recovered result by algorithms. .................................................. 31
6.11 Sample recovery results for Lenna and Pepper images. From top row to bottom row, 60%, 70% and 80% of tensor values are missing for each image respectively. .................................................. 32
1.1 Problem

In today’s information-rich environment, personalized recommendations have become increasingly important for information retrieval and content discovery. By giving suggestions that are relevant to users’ needs and interests, they allow users to navigate the information efficiently. For instance, think of Amazon product recommendation, Youtube video recommendation, and Netflix movie recommendation. What is common in those recommendation systems is that we observe ratings from each user for some set of items. Based on user’s past behavior, relationship to other users, item similarity, etc., the system aims to estimate the unobserved ratings for some user-item pairs. Such predictions allow the system to provide suggestions for items that are most likely to be of interest to a particular user.

More formally, in the Matrix Completion problem, we represent each user \( u \) as an element in \( \mathbb{R} \), and similarly for each item. There are \( n \) users and \( m \) items. And we have \( N \) total ratings \( \mathcal{D} = \{ y(u^s, i^s) \in \mathbb{R} \}_{s=1}^{N} \), captured in a \( n \times m \) rating matrix \( A \). The problem is to infer and fill the missing entries in \( A \) based on the subset of known entries. Two main classes of methods exist for predicting unknown ratings: collaborative filtering algorithms [6] and matrix factorization based approaches [14]. Collaborative filtering analyzes the relationship between users or products, and uses the known preferences of a group of users to predict the unknown preference of new users. Matrix factorization, on the other hand, assumes that the matrix is low-rank and attempts to capture the underlying phenomenon by approximating the matrix as the product of matrices.
CHAPTER 1. INTRODUCTION

1.2 Contributions

The main contribution of this thesis is to introduce the framework of “Blind Regression” and show its applicability for the problem of matrix completion as well as as tensor completion.

In our work, we assume that there are some latent features $x_1(u) \in \mathcal{X}_1$ associated with user $u \in [n]$, and similarity latent features $x_2(i) \in \mathcal{X}_2$ for item $i \in [m]$. We also assume that the rating $y(u,i) = f(x_1(u), x_2(i)) + \eta_{u,i}$, where $f : \mathcal{X}_1 \times \mathcal{X}_2 \to \mathbb{R}$ is a function that captures the structure in the ratings and $\eta_{u,i}$ are drawn from i.i.d. normal random variables representing the noise. In classical regression, the features $x_1(u)$ and $x_2(i)$ are observed and therefore the function $f$ could be directly learned for making future predictions. However, in our case, we do not observe latent features but instead only the unique identifiers indexing the data points. Thus we term our framework “Blind Regression” and develop a nonparametric method to fill in missing entries.

Our method operates in two steps to predict an unknown entry $y(u,i)$. First, from each neighbor $(v,j)$, it produces a basic estimate

$$y(u,i) \approx y(v,i) + y(u,j) - y(v,j).$$

The prediction is derived from the classical Taylor approximation of a function.

For the second step, the method combines and weighs the preferences of those neighbors. From the Taylor approximation, we know that if $x_1(v)$ is near $x_1(u)$, $x_2(j)$ is near $x_2(i)$ and the function $f$ is not unreasonably behaved (e.g. Lipschitz continuous), the local approximation will be accurate. As the latent features are not observed, we instead calculate the sample variance of the rating difference between pairs of users or items. We use this empirical variance as a proxy for the distance between $x_1(u)$ and $x_1(v)$, or between $x_2(i)$ and $x_2(j)$. Our error analysis shows that sample variance of the differences used for weighing the basic estimates provides empirical confidence interval for our prediction, and experimental evaluations also suggest that it is a good proxy of the prediction error in the estimates.

It is not hard to see that our algorithm is similar to the classical collaborative filtering methods, which weigh the local estimates from individual neighbors by the similarity between them. In fact, as discussed in chapter 4, the basic version of our algorithm which we analyze is essentially the nearest-neighbor user-user or item-item collaborative filtering. Therefore our analysis provides theoretical understanding of the
collaborative filtering algorithm and we prove that the power of collaborative filtering is that it performs nonparametric learning for any underlying Lipschitz function. In addition, we present a variant of our algorithm and the experiments on real-world data sets demonstrate its performance improvement over classic collaborative filtering algorithms in terms of prediction accuracy.

Finally, because we assume generic conditions on the model and the function $f$, our framework easily extends to the setting of higher-order tensors. This suggests the general applicability and flexibility of our Blind Regression framework.

\section*{1.3 Thesis Organization}

The remainder of this thesis is structured as follows. Chapter 2 introduces the related literature on matrix completion as well as tensor completion. In chapter 3, we define our model and assumptions, the notations and the performance metrics used in the experiments. Chapter 4 presents our algorithm for matrix completion including its two variants and explains how it could be extended to tensor completion. Chapter 5 provides sample complexity analysis and error analysis of our matrix completion algorithm. Chapter 6 details the experimental evaluations of our method for matrix completion on data from the MovieLens and Netflix challenge, showing that it improves over user-user and item-item collaborative filtering methods. We also evaluated the performance of our tensor completion algorithm on image inpainting problem and the results suggest that it is competitive with existing state-of-the-art tensor completion heuristics. Finally, we conclude and provide directions for future work in chapter 7.
Chapter 2

Background

In this chapter, we discuss the related literature on matrix completion and tensor completion. We also provide a brief introduction to kernel regression in the end.

2.1 Matrix Completion

In 2006, Netflix Inc started the Netflix Prize competition that rewards the team who could substantially improve the accuracy of predictions for movie recommendation. The prize, combined with the availability of the customer generated movie ratings data set, led to great academic interest on the problem and the advancement of techniques for recommendation systems. Below, we introduce two classes of methods commonly used in the recommendation systems.

2.1.1 Collaborative Filtering

One class of method is collaborative filtering (CF). It analyzes the relationship between users and utilizes the data/preferences from other users to make better recommendation. One common form of collaborative filtering method is neighborhood-based in which predictions are made based on the data from other users whose ratings are similar to that of target user, or from other items whose ratings are similar to that of target item. For example, in user-user CF approach, the predicted rating for user $u$ on item $i$, $\hat{y}(u,i)$, is the convex combination of the neighboring user’s ratings, where the weights are chosen proportional to the cosine similarity of the pair of users. If we let $sim(u,v)$ denote the similarity between user $u$ and $v$, the precise expression for the user-user collaborative filtering estimates is given by

$$
\hat{y}(u,i) = m_u + \frac{\sum_{v \in N_2(i) \setminus \{u\}} sim(u,v)(y(v,i) - m_v)}{\sum_{v \in N_2(i) \setminus \{u\}} |sim(u,v)|}.
$$
Similarly, in item-item CF approach, item-item similarity is used and the method estimates the rating as the weighted average of the ratings user $i$ gives to other movies. The corresponding expression for the item-item CF estimates is given by

$$
\hat{y}(u, i) = m_i + \frac{\sum_{j \in N_1(u) \setminus \{i\}} sim(i, j)(y(u, j) - m_j)}{\sum_{j \in N_1(u) \setminus \{i\}} |sim(i, j)|}.
$$

In above $m_u$ and $m_i$ denote the mean values of entries for row of user $u$ and column of item $i$, respectively. $N_1(u)$ denotes the set of items rated by user $u$ and $N_2(i)$ denotes the set of users who have rated item $i$ (see chapter 3).

Two choices for similarity are Pearson correlation and cosine similarity. For instance, the cosine similarity of two users $u$ and $v$ is defined as

$$
sim(u, v) = \frac{\sum_{j \in N_1(u, v)} y(u, j)y(v, j)}{\sqrt{\sum_{j \in N_1(u)} y(u, j)^2} \sqrt{\sum_{j \in N_1(v)} y(v, j)^2}}
$$

where $N_1(u, v)$ denotes the set of items rated by both user $u$ and user $v$.

CF methods have proven to be successful in practice and are widely applied in commercial recommendation systems [8, 15, 22]. Yet there is little theoretical understanding of why they perform well. In [3], the authors introduce latent source models for online collaborative filtering to analyze the performance of a cosine-similarity collaborative filtering method. The model imposes a structure over users by assuming that they come from a relatively small number of latent sources.

### 2.1.2 Matrix Factorization

Another class of method is matrix factorization based which assumes that there are some latent features underlying the interactions between users and items. The model then tries to map users and items to this joint latent factor space of reduced dimensionality $r$ by finding matrix $P \in \mathbb{R}^{n \times r}$ and $Q \in \mathbb{R}^{m \times r}$ such that $A \approx P \times Q^T$. In this way, $u$th row of $P$ measures the strength of the connection between user $u$ and the underlying features. Similarly, $i$th row of $Q$ measures the strength of the relationship between item $i$ and the features. The estimate of user $u$’s rating on item $i$ is then given
by the dot product of the two factor vectors $p_u, q_i$ corresponding to user $u$ and item $i$:

$$\hat{y}(u, i) = p_u^T q_i = \sum_{k=1}^{r} p_{uk} q_{ki}.$$ 

To find $P$ and $Q$, the method minimizes the regularized square error between the estimated ratings and the true ratings. Such optimization problem can be solved using stochastic gradient descent and alternating least squares [9, 14].

Compared with CF methods, matrix factorization methods generally have smaller sample complexities [4, 12]. However, the required low-rank assumption might not hold true in practice and can be easily violated. As observed in [5], a simple non-linear transformation over the matrix entries could render it full rank. Unlike low-rank matrix factorization, our model assumptions are less restrictive as we allow for general Lipschitz continuous functions. In fact, low-rank matrix factorization becomes an instance our blind regression framework if we simply constrain the function to be specifically the inner product between the latent representations.

\section*{2.2 Tensor Completion}

A tensor is a high dimensional analogs of matrices. Compared with matrix completion, tensor completion is more difficult and algorithms that are well-established and developed for matrices can not be easily extended and analyzed for tensors [13]. Despite those challenges, many tensor factorization techniques have been developed, among them CANDECOMP/PARAFAC (CP) that decomposes a tensor as a sum of rank-one tensors [7]. There is also theoretical work analyzing the sample complexity for tensor factorization [10].

In terms of application, tensor decomposition is used in a number of fields such as computer vision, e.g., to estimate missing values in visual data [16, 23] and analyze facial images [20]. It has also proven to be useful in learning Latent variable models. For instance, moments of data which can be estimated empirically are naturally represented by tensors and the low rank decomposition of such tensors can be used to infer the parameters of the underlying model [1].

\section*{2.3 Kernel Regression}

Kernel regression [17, 21] is a non-parametric technique to estimate the conditional expectation of a random variable. Given $n$ input data pairs $(x_1, y_1), \ldots, (x_n, y_n)$, it
attempts to estimate $E[Y|X = x]$ by fitting a $p$th order local polynomial at $x$. The estimator weighs each local observable point by a kernel function $K_h(\cdot)$. This kernel basis function depends on the distance from that point to $x$ and is controlled through bandwidth $h$.

If $p = 0$, we have the Nadaraya-Watson estimator [2]:

$$\hat{m}(x) = \frac{\sum_{i=1}^{n} K_h(x - x_i)y_i}{\sum_{i=1}^{n} K_h(x - x_i)}$$

The asymptotic correctness of kernel regression is proved through Taylor’s theorem. Taking the idea from kernel regression, our algorithm produces local estimate using Taylor’s approximation, and a variant of our algorithm weighs each local estimate by an exponentially decaying kernel function. However, as the latent features are not observable in our setting, we need to estimate the distance between $x$ and $x_i$ and we use the sample variance of the difference in observed ratings between a pair of users or items as a proxy for this distance.
Chapter 3

Setting

In this chapter, we present our model and detail the assumptions made by it. We also introduce the notations and define the performance metrics we use to compare our methods with existing approaches in the experiments.

3.1 Model and Assumptions

As discussed in Chapter 1, there are $n$ users and $m$ items. For every $u \in [n]$ and $i \in [m]$, there is a true corresponding latent type representation, $x_1(u) \in \mathcal{X}_1$, and $x_2(i) \in \mathcal{X}_2$. And the rating of user $u$ for item $i$, denoted by $y(u, i)$ is given by $y(u, i) = f(x_1(u), x_2(i)) + \eta_{u,i}$. These ratings are captured in a rating matrix $A \in \mathbb{R}^{n \times m}$, with $A(u, i) = y(u, i)$ if the rating from user $u$ for item $i$ is available and 0 otherwise.

For the purposes of the analysis of our algorithm, we assume the following properties.

(a) $\mathcal{X}_1$ and $\mathcal{X}_2$ are bounded metric spaces endowed with norm $\| \cdot \|_{\mathcal{X}_1}$ and $\| \cdot \|_{\mathcal{X}_2}$ respectively:

$$\|x_1 - x'_1\|_{\mathcal{X}_1} \leq B_{\mathcal{X}} \quad \forall \ x_1, x'_1 \in \mathcal{X}_1,$
$$

and

$$\|x_2 - x'_2\|_{\mathcal{X}_2} \leq B_{\mathcal{X}} \quad \forall \ x_2, x'_2 \in \mathcal{X}_2.$$

(b) $f : \mathcal{X}_1 \times \mathcal{X}_2 \rightarrow \mathbb{R}$ is $L$-Lipschitz:

$$|f(x_1, x_2) - f(x'_1, x'_2)| \leq L(\|x_1 - x'_1\|_{\mathcal{X}_1} + \|x_2 - x'_2\|_{\mathcal{X}_2}), \forall x_1, x'_1 \in \mathcal{X}_1, x_2, x'_2 \in \mathcal{X}_2.$$

(c) $P_{\mathcal{X}_1}$ and $P_{\mathcal{X}_2}$ are the probability measures on $(\mathcal{X}_1, T_{\mathcal{X}_1})$ and $(\mathcal{X}_2, T_{\mathcal{X}_2})$ respectively, where $T_{\mathcal{X}}$ denotes the Borel $\sigma$-algebra of a metric space $\mathcal{X}$. We assume that $x_1(u)$, the latent feature of each user $u$, and $x_2(i)$, the latent feature of each item $i$, are sampled independently with distribution given by $P_{\mathcal{X}_1}$ and $P_{\mathcal{X}_2}$ respectively.
(d) The additive noise for all entries are independent with mean zero and variance $\gamma^2$:

$$E[\eta_{u,i}] = 0, \quad \text{Var}[\eta_{u,i}] = \gamma^2, \quad \forall u \in [n], \ i \in [m].$$

(e) Rating of each entry is revealed independently with probability $p$.

### 3.2 Notations

We first define a random variable $M(u, i) = 1$ iff rating from user $u$ for item $i$ is observed and 0 otherwise, i.e., $M(u, i)$ is a Bernoulli random variable with parameter $p$.

Let $N_1(u)$ denote the set of items rated by user $u$, i.e., the set of column indices of revealed entries in row $u$. Similarly, let $N_2(i)$ denote the set of ratings received by item $i$. More formally,

$$N_1(u) \triangleq \{i : M(u, i) = 1\} \quad \text{and} \quad N_2(i) \triangleq \{u : M(u, i) = 1\}. \quad (3.1)$$

For two rows $u \neq v$, define $N_1(u, v)$ to be the set of column indices of commonly observed entries of both rows, $N_1(u) \cap N_1(v)$. Similarly, for two columns $i \neq j$, define $N_2(i, j)$ to be the set of row indices of commonly observed entries of both columns, $N_2(i) \cap N_2(j)$. We will refer to this as the overlap between two rows or columns.

Finally, let $S^\beta_u(i)$ denote all other users who have rated item $i$ and have at least $\beta$ commonly rated items with user $u$. And let $S^\beta_i(u)$ denote all other items user $u$ has rated except $i$ for which there are at least $\beta$ users who have rated both $i$ and $j$. That is,

$$S^\beta_u(i) = \{v \ s.t. \ v \in N_2(i), \ v \neq u, \ |N_1(u, v)| \geq \beta\}, \quad (3.2)$$

$$S^\beta_i(u) = \{j \ s.t. \ j \in N_1(u), \ j \neq i, \ |N_2(i, j)| \geq \beta\}. \quad (3.3)$$

### 3.3 Evaluation Metrics

In this section, we define the performance metric that we use to quantify the performance of our prediction algorithm in the experiments. Let $\hat{A} \in \mathbb{R}^{n \times m}$ be the output of algorithm, i.e., input matrix $A$ with missing entries filled. And let $E \subset [n] \times [m]$ be the set of unobserved user-movie indices for which the algorithm predicts a rating. Specifically, define $\hat{y}(u, i)$ to be the predicted rating while the true (unknown) rating is
y(u, i) for (u, i) ∈ E. We have the following metrics:

- **Mean Absolute Error (MAE)**
  \[
  MAE = \frac{1}{|E|} \sum_{(u,i) \in E} |y(u, i) - \hat{y}(u, i)|.
  \]

- **Root Mean Square Error (RMSE)**
  \[
  RMSE = \sqrt{\frac{1}{|E|} \sum_{(u,i) \in E} (y(u, i) - \hat{y}(u, i))^2}.
  \]

  Compared with MAE, RMSE penalizes large errors more heavily. And lower MAE and RMSE indicate higher prediction accuracy.

- **Relative Square Error (RSE)**
  \[
  RSE = \frac{\|\hat{A} - A\|_F}{\|A\|_F}
  \]
  where \(\|\cdot\|_F\) is the Frobenius norm of the matrix. RSE can be seen as the normalized RMSE.

- **Kendall-Tau Distance**: For each missing entry \(A(u, i)\), we check if the preferences between item \(i\) and each of the other items rated by user \(u\) are changed by the estimation \(\hat{y}(u, i)\). Kendall-Tau Distance calculates the percentage of pairwise disagreements among all possible pairs.

  \[
  Distance = \frac{\sum_{u=1}^{n} \sum_{i \notin N_1(u), j \in N_1(u)} \text{sgn}(y(u, i) - y(u, j)) \neq \text{sgn}(\hat{y}(u, i) - y(u, j))}{\sum_{u=1}^{n} |N_1(u)|(m - |N_1(u)|)}
  \]
Chapter 4

Algorithm

In this chapter, we present the algorithms obtained by applying our Blind Regression framework to the problem of matrix completion and tensor completion. We start with a brief explanation of intuition behind how our algorithm produces the basic estimate for an unknown entry from a neighbor. We then provide details on the algorithm and its variants, and also explain the connection of our algorithm to the collaborative filtering methods.

4.1 Intuition

Consider the case where we know \( f(x_1(v), x_2(j)), f(x_1(v), x_2(i)), f(x_1(u), x_2(j)) \) but not \( f(x_1(u), x_2(i)) \), how could we make predictions for \( f(x_1(u), x_2(i)) \)?

Observe that

\[
f(x_1(u), x_2(j)) = f(x_1(v), x_2(j)) + \frac{\partial f(x_1(v), x_2(j))}{\partial x}(x_1(u) - x_1(v)) + o(x_1(u) - x_1(v))
\]

\[
f(x_1(v), x_2(i)) = f(x_1(v), x_2(j)) + \frac{\partial f(x_1(v), x_2(j))}{\partial y}(x_2(i) - x_2(j)) + o(x_2(i) - x_2(j))
\]

by Taylor’s theorem.

Applying Taylor’s theorem once more on \( f(x_1(u), x_2(i)) \) and substituting the above approximations, we have
\[ f(x_1(u), x_2(i)) \approx f(x_1(v), x_2(j)) \]
\[ + \frac{\partial f(x_1(v), x_2(j))}{\partial y}(x_2(i) - x_2(j)) + \frac{\partial f(x_1(v), x_2(j))}{\partial x}(x_1(u) - x_1(v)) \]
\[ + \frac{1}{2}[\frac{\partial^2 f(x_1(v), x_2(j))}{\partial x^2}(x_1(u) - x_1(v))^2 + \frac{\partial^2 f(x_1(v), x_2(j))}{\partial y^2}(x_2(i) - x_2(j))^2] \]
\[ + \frac{\partial^2 f(x_1(v), x_2(j))}{\partial x \partial y}(x_1(u) - x_1(v))^2(x_2(i) - x_2(j))^2 \]
\[ \approx f(x_1(v), x_2(j)) + f(x_1(v), x_2(i)) - f(x_1(v), x_2(j)) \]
\[ + f(x_1(u), x_2(j)) - f(x_1(v), x_2(j)) \]
\[ = f(x_1(v), x_2(i)) + f(x_1(u), x_2(j)) - f(x_1(v), x_2(j)) \]

Thus if the noise is small, an unknown rating for user \( u \) and item \( i \) can be predicted using known ratings \( y(u, j) \), \( y(v, i) \) and \( y(v, j) \) according to

\[ \hat{y}(u, i) = y(u, j) + y(v, i) - y(v, j). \] (4.1)

If \( x_1(u) \) is close to \( x_1(v) \) and \( x_2(i) \) is close to \( x_2(j) \), the above estimate is accurate.
Since in our setting, the features are not observed, we measure the proximity through empirical observations. Specifically, we compute the sample variance of the difference in ratings between pair of rows or columns to determine their proximity. As described below, our algorithm weighs each basic estimate according to the computed distance and combines these basic estimates to make final prediction for \( \hat{y}(u, i) \).

### 4.2 Matrix Completion Algorithm

**Algorithm.** The algorithm uses parameter \( \beta \geq 2 \). \( \beta \) sets the minimal overlap required between rows or columns in order to use the ratings in them for prediction.

To predict a rating for entry \( (u, i) \):

1. Compute the set of relevant entries

\[ S_u^\beta(i) = \{ v \text{ s.t. } v \in N_2(i), \ v \neq u, \ |N_1(u, v)| \geq \beta \}, \] (4.2)
\[ S_i^\beta(u) = \{ j \text{ s.t. } j \in N_1(u), \ j \neq i, \ |N_2(i, j)| \geq \beta \}. \] (4.3)
2. For all \( v \in S_u^\beta(i) \), compute empirical row mean and variances between \( u \) and \( v \):

\[
m_{uv} = \frac{1}{|N_1(u,v)|} \sum_{j \in N_1(u,v)} (y(u,j) - y(v,j))
\]

\[
s_{uv}^2 = \frac{1}{|N_1(u,v)| - 1} \sum_{j \in N_1(u,v)} \left( (y(u,j) - y(v,j)) - m_{uv} \right)^2.
\]

(4.4) (4.5)

Similarly, for each \( j \in S_i^\beta(u) \), compute empirical column mean and variances between \( i \) and \( j \):

\[
m_{ij} = \frac{1}{|N_2(i,j)|} \sum_{v \in N_2(i,j)} (y(v,i) - y(v,j))
\]

\[
s_{ij}^2 = \frac{1}{|N_2(i,j)| - 1} \sum_{v \in N_2(i,j)} \left( (y(v,i) - y(v,j)) - m_{ij} \right)^2.
\]

(4.6) (4.7)

3. Compute the weights \( w_{vj} \) from (4.5) and (4.7) for the estimate of \( y(u,i) \) based on entries \( y(v,j), y(u,j), y(v,i) \). The two following variants of our algorithm differ in the way these weights are chosen.

4. Predict missing entry in position \((u,i)\) as

\[
\hat{y}(u,i) = \frac{\sum_{(v,j) \in B^\beta(u,i)} w_{vj} (y(u,j) + y(v,i) - y(v,j))}{\sum_{(v,j) \in B^\beta(u,i)} w_{vj}},
\]

where \( B^\beta(u,i) = \{(v,j) \in S_u^\beta(i) \times S_i^\beta(u) \text{ s.t. } M(v,j) = 1\} \).

Variation 1. User-User Nearest Neighbor

Instead of using all neighbors for estimation, we find the row with the minimum sample variance

\[
u^* \in \arg \min_{v \in S_u^\beta(i)} s_{uv}^2
\]

and include only basic estimates from that row. This is equivalent to set weights to 1 for all entries in row \( u^* \) and 0 otherwise. So

\[
w_{vj} = I(v = u^*).
\]

We provide sample complexity and error analysis for this variant of our algorithm.
in chapter 5.

**Variation 2. User-Item Gaussian Kernel Weights**

In this variant of our algorithm, we follow the idea from Kernel regression and weigh each estimate using a Gaussian kernel. The kernel values, controlled through a bandwidth parameter $\lambda$, are computed from the sample variance. More specifically, the unnormalized weight for the estimate of $y(u, j)$ based on entries $y(v, j), y(v, i)$, and $y(u, j)$ is given by

$$w_{vj} = \exp(-\lambda \min\{s_{uv}, s_{ij}\}).$$

When $\lambda = \infty$, the estimate only depends on the basic estimates whose row or column has the minimum sample variance (similar to Variation 1). When $\lambda = 0$, the neighbors are equally important and the algorithm simply averages all basic estimates.

We implement this variant of our algorithm and compare it with existing matrix completion methods in chapter 6.

**Connections to Collaborative Filtering**

Our algorithm is closely related to the classical collaborative filtering methods.

In neighborhood-based CF method, the prediction is produced by taking the weighted average of all the ratings with the weights being the similarity between two users or items. Similarly in our method, we use the idea from kernel regression and combine the local estimates based on the distance between the neighbors. As the latent representation $(x_1(u), x_2(i))$ for $(u, i)$ is unobserved, we cannot directly calculate $\|x_1(u) - x_1(v)\|$ or $\|x_2(i) - x_1(j)\|$. Instead, we use sample variance of the difference between pair of users or items as a substitute for the distance to compute final weights. In some sense, we can view the cosine similarity metric in nearest neighbor methods as an approximate for the kernel function in kernel regression.

In addition, variation 1 of our algorithm is asymptotically the same as the mean-adjusted nearest-neighbor CF, which gives the estimate by $\hat{y}(u, i) = m_u + y(u^*, i) - m_u^*$, where $m_u$ and $m_u^*$ are the sample averages of these rows. As shown in chapter 5, we can simplify the estimate given by variation 1 of our algorithm as

$$\hat{y}(u, i) = y(u^*, i) + m_{uu^*}.$$

When the number of overlap $|N_1(u, u^*)|$ is large, $m_{uu^*} \approx m_u - m_{u^*}$, such that the two estimates are equivalent. Thus our analysis directly implies the proof of correctness for the classic nearest-neighbor CF method.
4.3 Extension to Higher Dimensions

Our method could be extended to the setting of higher dimension tensors easily and here we present three different ways of applying our method to solve tensor completion problem for third order tensors. Specifically, we consider a third-order tensor of size $n \times m \times r$, with latent spaces $X_1, X_2, X_3$.

**Approach 1**
The most naive way is to carry out our matrix completion algorithm on each slices of the tensor along any of the three coordinates.

**Approach 2**
We could flatten the tensor to a $n \times mr$ matrix with latent spaces $X_1$ and $X_2 \times X_3$ (the product spaces over $X_2$ and $X_3$). Assume that each index $u \in [n], i \in [m], k \in [r]$ is associated to a latent feature $x_1(u), x_2(i), x_3(k)$ sampled independently from bounded space respectively. We also assume that $y(u, i, k) = f(x_1(u), x_2(i), x_3(k) + \eta_{u,i,k}$ where $f : X_1 \times X_2 \times X_3 \rightarrow \mathbb{R}$ is some $L$-Lipschitz function that relates latent features to the observed values and $\eta_{u,i,k}$ are independent noise with zero mean and variance $\gamma^2$. Since the assumptions required by our matrix completion algorithm still hold, we apply it over the “flattened” matrix.

**Approach 3**
Similar to how the basic estimate for an unknown entry is derived for matrix completion, we can rederive the local estimator for higher dimension tensor completion using local Taylor approximations. We have

$$f(x_1(u), x_2(i), x_3(k)) \approx f(x_1(v), x_2(j), x_3(l)) + (x_1(u) - x_1(v)) \frac{\partial f(x_1(v), x_2(j), x_3(l))}{\partial x_1(v)}$$

$$+ (x_2(i) - x_2(j)) \frac{\partial f(x_1(v), x_2(j), x_3(l))}{\partial x_2(j)}$$

$$+ (x_3(k) - x_3(l)) \frac{\partial f(x_1(v), x_2(j), x_3(l))}{\partial x_3(l)}$$

$$\approx f(x_1(v), x_2(j), x_3(l)) + (f(x_1(u), x_2(j), x_3(l)) - f(x_1(v), x_2(j), x_3(l)))$$

$$+ (f(x_1(v), x_2(i), x_3(l)) - f(x_1(v), x_2(j), x_3(l)))$$

$$+ (f(x_1(v), x_2(j), x_3(k)) - f(x_1(v), x_2(j), x_3(l)))$$
This suggests that the 3-dimensional basic estimate is obtained by

\[
\hat{y}(u, i, k) = y(v, j, l) + [y(u, j, l) - y(v, j, l)] \\
+ [y(v, i, l) - y(v, j, l)] + [y(v, j, k) - y(v, j, l)] \\
= y(u, j, l) + y(v, i, l) + y(v, j, k) - 2y(v, j, l)
\]

The weights can be computed from estimating the distances in the latent space via a squared difference between overlapping entries when varying one coordinate and keeping other coordinates constant, i.e., computing the distance between two slices. This extension provides a principled framework to deal with the problem of tensor completion.
Chapter 5

Analysis

In this chapter, we analyze variant 1 of our algorithm. We give the sample complexity of our algorithm, and provide a tail bound for the probability of the estimation error based on the minimum sample variance from data.

5.1 Sample Complexity Analysis

We first analyze the sample complexity of our algorithm and show that our algorithm is feasible, i.e., we are able to give the estimate for an known entry $y(u, i)$ with high probability. Recall that our algorithm predicts $y(u, i)$ by first choosing the neighboring row with the minimum sample variance, $u^* = \arg\min_{v \in S_u^\beta(i)} s_{uv}$, therefore feasibility requires that $|S_u^\beta(i)| \geq 1$. The following lemma shows that in fact, $|S_u^\beta(i)|$, the number of the candidate rows, concentrates around $(n - 1)p$ with high probability. This relies on concentration of Binomial random variables via Hoeffding’s inequality.

**Lemma 5.1.1.** For any $\beta < mp^2, \alpha > 0, u \in [n], i \in [m],$

$$
P \left( \left| S_u^\beta(i) \right| - (n - 1)p \geq \alpha \right) \leq (n - 1) \exp \left( \frac{-2(\beta - mp^2)^2}{m} \right) + 2 \exp \left( \frac{-2\alpha^2}{n - 1} \right).$$

**Proof.** Recall that we define the set $S_u^\beta(i)$ as

$$S_u^\beta(i) = \{ v \ s.t. \ v \in N_2(i), \ v \neq u, \ |N_1(u, v)| \geq \beta \},$$

i.e., $S_u^\beta(i)$ consists of all neighboring rows $v$ such that $y(v, i)$ is observed, and the overlap between row $v$ and $u$ is at least than $\beta$.

We know that the event that all rows have overlap with $u$ at least $\beta$, and the number of observed entries in column $i$ is inside the interval of $\alpha$ around $(n - 1)p$ implies $\left| S_u^\beta(i) \right| - (n - 1)p \leq \alpha$. 

19
By taking the contrapositive, we have that the event $|S_u(i)^\beta| - (n-1)p \geq \alpha$ implies that either (a) there exists at least one row whose overlap with $u$ is less than $\beta$, or (b) $|N_2(i) \setminus u|$ is outside the interval of $\alpha$ around $(n-1)p$. So

\[
\mathbb{P} \left( |S_u(i)^\beta| - (n-1)p \geq \alpha \right) \leq \mathbb{P} \left( \bigcup_{v \in [n] \setminus u} \{ |N_1(u,v)| < \beta \} \right) + \mathbb{P} (|N_2(i) \setminus u| - (n-1)p \geq \alpha) \\
\leq \sum_{v \in [n] \setminus u} \left( \mathbb{P} (|N_1(u,v)| < \beta) \right) + \mathbb{P} (|N_2(i) \setminus u| - (n-1)p \geq \alpha)
\]

by union bound.

We observe that $|N_1(u,v)|$ is distributed as a Binomial($m,p^2$), and $|N_2(i) \setminus u|$ is distributed as a Binomial($(n-1),p$). By applying Hoeffding’s inequality for Binomial random variables, it follows that

\[
\mathbb{P} \left( |S_u(i)^\beta| - (n-1)p \geq \alpha \right) \leq (n-1) \exp \left( -\frac{2(\beta - mp^2)^2}{m} \right) + 2 \exp \left( -\frac{2\alpha^2}{n-1} \right).
\]

The number of feasible base rows $|S_u(i)^\beta|$ concentrates to $(n-1)p$ when $p$ satisfies $(n-1)p > 0$, and $\alpha, \beta$ are chosen so that $(n-1)p \gtrsim \alpha$ and the right-hand side = $o(1)$. If $p = \omega(m^{-1/4})$ and $p = \omega(n^{-1/2})$, taking $\alpha = \frac{(n-1)p}{2}$ and $\beta = \frac{mp^2}{2}$ fulfills the conditions.

\section{5.2 Error Analysis}

In the following section, we provide the tail bound for the probability of the estimation error of our algorithm based on empirical version of Chebyshev’s inequality.

**Upper bound the probability of the estimation error**

**Proposition 5.2.1.** For any $u \in [n], i \in [m]$, and $\epsilon > 0$,

\[
\mathbb{P} \left( |y(u,i) - \hat{y}(u,i)| > \epsilon \right) \leq \frac{s_{uu}^2}{\epsilon^2} + \frac{1}{\beta}
\]
Proof. Note that our estimation error is

\[
y(u, i) - \hat{y}(u, i) = y(u, i) - \frac{1}{|N_1(u, u^*)|} \sum_{j \in N_1(u, u^*)} (y(u^*, i) + y(u, j) - y(u^*, j))
\]

\[
= \frac{1}{|N_1(u, u^*)|} \sum_{j \in N_1(u, u^*)} (y(u, i) - (y(u^*, i) + y(u, j) - y(u^*, j)))
\]

\[
= \frac{1}{|N_1(u, u^*)|} \sum_{j \in N_1(u, u^*)} ([y(u, i) - y(u^*, i)] - [y(u, j) - y(u^*, j)])
\]

\[
= (y(u, i) - y(u^*, i)) - \frac{1}{|N_1(u, u^*)|} \sum_{j \in N_1(u, u^*)} (y(u, j) - y(u^*, j))
\]

\[
= (y(u, i) - y(u^*, i)) - m_{uu^*}
\]

Let’s assume that \( P(y(u, i) - y(u^*, i) = y(u, j) - y(u^*, j)) = 0, \forall j \in N_1(u, u^*). \) Since \( y(u, i) - y(u^*, i) \) and \( y(u, j) - y(u^*, j), j \in N_1(u, u^*) \) are exchangeable samples, we could apply the following lemma. With \( \lambda = \frac{\epsilon}{Q} = \sqrt{|N_1(u, u^*)|} / (|N_1(u, u^*)| + 1) \sqrt{s_{uu^*}^2}, \) we have

\[
P\left( |y(u, i) - \hat{y}(u, i)| > \epsilon \right) \leq \frac{|N_1(u, u^*)|^2 - 1}{|N_1(u, u^*)|^2} \left( \frac{s_{uu^*}^2}{\epsilon^2} + \frac{1}{|N_1(u, u^*)|} \right)
\]

\[
\leq \frac{s_{uu^*}^2}{\epsilon^2} + \frac{1}{|N_1(u, u^*)|}
\]

\[
\leq \frac{s_{uu^*}^2}{\epsilon^2} + \frac{1}{\beta}
\]

We note that as the error bound depends on the minimum sample variance achieved from data and this gives us a confidence interval about our estimation error.

**Lemma 5.2.1** (Empirical Chebyshev’s inequality [11, 19]). Let \( n \geq 2 \) a fixed integer, \( x_1, x_2, \ldots, x_n \) and \( y \) be weakly exchangeable samples from some unknown distribution such as \( P(x_1 = x_2 = \ldots = x_n = y) = 0. \) Denote \( \bar{X} := \frac{1}{n} \sum_{i=1}^{n} x_i \) and \( \text{Var}(X) := \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{X})^2 \) the sample mean and variance respectively, and let \( Q^2 = \frac{n+1}{n} \text{Var}(X) \). For \( \lambda \geq 1, \)

\[
P(|y - \bar{X}| \geq \lambda Q) \leq \frac{1}{n+1} \left[ \frac{n+1}{n} \left( \frac{n-1}{\lambda^2} + 1 \right) \right]
\]
Upper Bound True Variance

For any fixed $a, b \in \mathcal{X}_1$, let us denote the variance of the difference $f(a, x) - f(b, x)$ by

$$
\sigma_{ab}^2 \triangleq \text{Var}_{x \sim P_{\mathcal{X}_2}}[f(a, x) - f(b, x)].
$$

When we condition on the latent representations of the users $x_1(u) = a, x_1(v) = b$, we can show that $\sigma_{ab}^2$ is equivalent to the expectation of the empirical variances computed plus the independent noises.

$$
E[s_{uv}^2|x_1(u) = a, x_1(v) = b] = \sigma_{ab}^2 + 2\gamma^2.
$$

As shown in the previous lemma, our error bound depends on the sample variance and in the following, we prove that the true variance of the difference across rows is bounded. This involves using the Lipschitz and bounded assumptions on $f$ and latent space $\mathcal{X}_1$ respectively.

**Lemma 5.2.2.** For any $u, v \in [n], j \in [m], Var_{x_2(j) \sim P_{\mathcal{X}_2}}[y(u, j) - y(v, j)] \leq 2L^2B^2 + 2\gamma^2$.

**Proof.** First by the Lipschitz property of $f$, we have that $|f(a, x) - f(b, x)| \leq L\|a - b\|_{\mathcal{X}_1}, \forall a, b \in \mathcal{X}_1, x \in \mathcal{X}_2$. We can now bound $\sigma_{ab}^2$:

$$
\sigma_{ab}^2 = \text{Var}_{x \sim P_{\mathcal{X}_2}}[f(a, x) - f(b, x)]
= E[(f(a, x) - f(b, x))^2] - E[f(a, x) - f(b, x)]^2
\leq E[(L\|a - b\|_{\mathcal{X}_1})^2] + E[L\|a - b\|_{\mathcal{X}_1}]^2
= 2L^2\|a - b\|_{\mathcal{X}_1}^2
\leq 2L^2B^2.
$$

It follows that conditioned on $x_1(u) = x_1(u), x_1(v) = x_1(v)$,

$$
Var_{x_2(j) \in \mathcal{X}_2}[y(u, j) - y(v, j)] = \sigma_{x_1(u) x_1(v)}^2 + 2\gamma^2
\leq 2L^2B^2 + 2\gamma^2.
$$

\[\blacksquare\]
Chapter 6

Experiments

In this chapter we present the result of comparing our method with existing matrix completion heuristics on real world datasets. We find that our matrix completion algorithm outperforms classic user-user and item-item collaborative filtering approaches. In addition, the result from real-world application of image inpainting demonstrates that our tensor completion algorithm is competitive with state-of-the-art approaches in terms of predictive performance.

6.1 Matrix Completion Experiments

We implemented the variation 2 of our algorithm that uses a Gaussian kernel to combine all the estimates, and we evaluated its performance on two data sets, MovieLens 1M and Netflix. We compared our method with user-user collaborative filtering and item-item collaborative filtering, as well as matrix factorization-based method soft-Impute from [18]. We find that our algorithm performs better than CF approaches in terms of achieving a lower Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and Kendall-Tau Distance.

We also find that the sample variance of the neighboring row or column used for estimation is a good proxy of the prediction error in the estimates – thus suggesting that sample variance is indeed a good proxy to determine empirical confidence interval as suggested by our main result.

6.1.1 Data Sets and Algorithms

Data Sets
The MovieLens 1M data set contains about 1 million ratings by 6000 users of 4000 movies from the online movie recommendation service MovieLens. The Netflix data set consists of about 100 million movie ratings by 480,189 users of about 17,770 movies.
In both data sets, the ratings are integers from 1 to 5 and we randomly sample 2000 users and 2000 movies from each data set to create a smaller rating matrix. To evaluate the recommendation methods, a percentage of the known ratings is randomly selected and withheld for evaluation, and the remaining portion of the data set is used by the algorithm to compute the estimates. Specifically, we evaluated with 10%, 30%, 50%, and 70% hold out sets, such that the larger percentage evaluation set is also equivalent to smaller percentage of data available to the algorithm.

**Algorithms Considered**

We implemented the variation 2 of our algorithm, which combines all estimates together using a Gaussian kernel. We chose the overlap parameter $\beta = 2$ in order to make sure that the algorithm is able to compute an estimate for all missing entries. When $\beta$ is larger, the algorithm enforces rows (or columns) to have higher overlap in commonly rated items (or users). Although this increases the reliability of the estimates, it also reduces the fraction of entries for which the estimate is well defined, since there may be some rows or columns which do not have sufficient overlap in observed entries with others. We also optimize the $\lambda$ bandwidth parameter by evaluating with multiple values for $\lambda$ and choosing the value which minimizes the error.

The algorithms we compare with include classic mean-adjusted user-user and item-item collaborative filtering methods with cosine similarity (as introduced in chapter 2). We also compare with matrix factorization-based method softImpute, which iteratively replaces the missing elements in the matrix with those obtained from a soft-thresholded SVD.

### 6.1.2 Experimental Results

Figure 6.1 plots the root mean squared error (RMSE), mean absolute error (MAE), and Kendall-Tau Distance resulting from running our algorithm along with the classic collaborative filtering algorithms and softImpute on MovieLens data set with varying sizes of evaluation set. The results on Netflix data set are shown in Figure 6.2. The values are generated by averaging the RMSE over 100 realizations of the subsampled dataset. The results suggest that our algorithm outperforms other methods on the evaluation sets of all tested sizes for both the MovieLens data set, while achieving comparable performance with collaborative filtering approaches on the Netflix data set.

**Choice of $\lambda$**

The parameter $\lambda$ corresponds to the inverse of variance in Gaussian kernel, in other
Figure 6.1: Performance of Algorithms on MovieLens Data Set. \( \lambda \) values used by our algorithm are 2.8 (10%), 2.3 (30%), 1.7 (50%), 1 (70%).

Figure 6.2: Performance of Algorithms on Netflix Data Set. \( \lambda \) values used by our algorithm are 1.8 (10%), 1.7 (30%), 1.6 (50%), 1.5 (70%) for Netflix.

In other words, it determines how much the algorithm emphasizes “close” neighbors with small sample variances over other neighbors. When \( \lambda = 0 \), the algorithm naively computes the average over all estimates, possibly allow “distant” neighbors with large sample variances to bias the estimate. When \( \lambda = \infty \), the algorithm computes its estimate using only the closest neighbor with the minimum sample variance. This could also increase the noise and variance in the estimate, since it relies only on a few entries. This highlights the tradeoff between incorporating many datapoints into the estimate to reduce the noise through averaging, and emphasizing only the datapoints which seem
to be closer in behavior to the target user or movie.

Figure 6.3 plots the RMSE as a function of $\lambda$ for our algorithm applied to the MovieLens data set with 10% evaluation set. The figure shows that the performance of the algorithm first improves with increasing values of $\lambda$ and then worsens as $\lambda$ grows larger, with optimal $\lambda \approx 3$.

In the caption of Figure 6.1 and 6.2, we reported the optimal value of $\lambda$ for each size of the available data. We observe that when the percentage of ratings available to the algorithm decreases (i.e. the percentage of evaluation set increases), the optimal value of $\lambda$ decreases, indicating that the algorithm needs to widen its circle to include estimates with larger sample variance. This intuitively makes sense, since the algorithm can depend more heavily on close neighbors as the matrix becomes denser, but needs to gather estimates more widely when the data is sparse.

**Existence of Close Neighbors**

For each $(u, i)$ in the evaluation set (10%) for the MovieLens data set, we find the row $v$ with minimum sample variance ($\min_v s_{uv}^2$) while requiring overlap of at least 5 ($\beta = 5$). Figure 6.4 shows the distribution over the value of the minimum sample variances. Observe that the minimum sample variance $s_{uv}^2 \leq 0.8$ for more than 90% of the entries, showing that it is unlikely for a user to have a closest neighbor with high sample variance, indicating that there is sufficient information to obtain good estimates through neighbor methods.

We divided the entries into different buckets based on their minimum sample variances (intervals of width 0.1 as plotted in Figure 6.4). We computed the error for each bucket, with the prediction that for estimates such that the nearest neighbors have
larger sample variance, the error will also vary more widely. Recall that our algorithm computes the estimates by a weighted combination of many values, where the weights decay exponentially with the sample variance. Therefore, the minimum sample variance only indicates the lowest sample variance among all values incorporated into the estimate. However, we will see that the minimum sample variance indeed provides a good indication of the reliability of the estimate.

Figure 6.5: Variation of Squared Error across $\min_v s_{uv}^2$ buckets ($\beta = 5$). The red line plots the median, while the box shows the 25th and 75th percentile, and end of dashed lines extends to the most extreme data point within 1.5 interquartile range of the upper quartile.
Figure 6.5 plots the squared error \((\hat{y}(u, i) - y(u, i))^2\) for each bucket, and Figure 6.6 plots the variance of the prediction error \(|\hat{y}(u, i) - y(u, i)|\) in each bucket as a function of \(\min_v s_{uv}^2\). As predicted, the variance and variability of the prediction error indeed increase with the minimum sample variance, validating the theoretical prediction that the sample variance is an observable measure of the reliability of the estimate. In fact this is quite useful in practice, since this implies that in addition to computing estimates for the missing entries, our algorithm can provide a confidence for each estimate obtained through a function of the sample variance of the entries involved in computing the final estimate.

\section*{6.2 Tensor Completion Experiments}

We evaluated and compared the performance of our tensor completion algorithm with methods in literature through image inpainting problem. The results show that despite its simplicity, our tensor completion algorithm is competitive with existing tensor factorization based approaches.

Figure 6.7 shows the first set of three images we tested. Each of the images is represented by 3rd order tensor with dimensionality of rows \times columns \times RGB and size \(256 \times 256 \times 3\). Various percentage of pixels are randomly removed and we use algorithms to fill in those removed entries. We compared our method against fast low rank tensor completion (FaLRTC) [16], alternating minimization for tensor completion (TenAlt) [10], and fully bayesian CP factorization (FBCP) [23], which extends CANDECOMP/PARAFAC(CP) tensor factorization with automatic tensor rank determination.
Sec. 6.2. Tensor Completion Experiments

Figure 6.7: Images used in Tensor Completion experiments.

Figure 6.8: Performance comparison between different approaches of our tensor completion algorithm based on RSE vs missing rate.

We first compared the performance of different approaches of our tensor completion algorithm and figure 6.8 shows the result for Lenna image by plotting RSE achieved as a function of test set size. Results for Pepper and Facade images are similar. For all three approaches, we set overlap requirement $\beta$ to 2 since we find no significant improvement of performance as $\beta$ increases and lower value of $\beta$ allows us to utilize as many entries as possible. For approach 2, we collapse the last two dimensions of the tensor (columns and RGB) to reduce the image to a matrix, and apply our method with user-item Gaussian kernel weights. We find that this approach gives the best performance. Compared with carrying out matrix completion in each of the R,G,B slices (Approach 1) which only utilizes the values along two particular dimensions, it
uses information across all dimensions. Approach 3 performs worst out of all three, possibly because the noise adds up, leading to large variance for the prediction. Indeed, when we look at the values generated, approach 3 gives larger number of predictions outside the range of input values.

Figure 6.9 shows Relative Squared Error (RSE) achieved by different tensor completion algorithms on three images, demonstrating that our tensor completion method (Approach 2 flattening tensor to matrix) is competitive with state-of-the-art tensor factorization based methods. In particular, it outperforms other methods on Lenna and Pepper images with missing rate of around 70%, although its performance deteriorates especially when the missing rate is high, e.g. 90%. In addition, as our method is neighborhood based, it does not capture global structure as well as those factorization based methods. Sample recovery results are shown in figure 6.11 and figure 6.10.
Figure 6.10: Sample recovery results for Facade image. 50%, 70% and 90% of values are removed for prediction for each row respectively. The first image in each row shows the original image with missing values, while the rest is the recovered result by algorithms.
Figure 6.11: Sample recovery results for Lenna and Pepper images. From top row to bottom row, 60%, 70% and 80% of tensor values are missing for each image respectively.
In this work, we introduce the nonparametric framework of Blind Regression and present the algorithms resulted when applying it to the problem of matrix completion and tensor completion.

We assume that there are latent features associated with each user and item sampled from some bounded metric space, and that the ratings are generated according to some Lipschitz continuous function on the features. Since we do not observe the latent features but only unique identifiers indexing the data points, we approximate the distances in the latent spaces by comparing overlapping entries in rows or columns. In our algorithm, we specifically use the sample variance of the difference in ratings between pair of users or items to determine the reliability of local estimate. We provide a tail bound for the probability of the estimation error which depends on the minimum sample variance obtained from data. In our experiments, we also find that the minimum sample variance serves as an observable measure of the reliability of the estimate.

Our method is closely related to the classic collaborative filtering method, in which the local estimates from neighbors are combined according to the similarity between the user/item and its neighbors. For us, one variant of our algorithm uses Gaussian kernel weights for combining all estimates. Another variant simply uses only basic estimates from the row with minimum sample variance, i.e., the nearest neighbor, and we show that this variant is essentially the nearest-neighbor collaborative method. Therefore our analysis provides theoretical understanding of the collaborative filtering method and suggests that one possible explanation for its success is that it performs nonparametric learning for some underlying Lipschitz function.

We evaluated the performance of our matrix completion algorithm on the MovieLens and Netflix data sets. The result demonstrates that our algorithm achieves systemic improvement in prediction accuracy over classic user-user and item-item collaborative filtering methods. Experimental evaluations of our tensor completion algorithm on im-
age inpainting problem suggests that it is competitive with state-of-the-art approaches in terms of predictive performance.

Next, we point out possible directions for further work.

The error bound in our analysis depends on the minimum sample variance. It is possible to show that the sample variance concentrates around the true variance and that there exists a neighbor whose true variance is small. Thus it is likely that the true variance between the nearest neighbor found by the algorithm is small and we can give an error bound that does not rely on the data.

Another possible direction of extension is to consider the setting in which the users or items have some observables features. For instance, in movie recommendation, we already know movie’s genre, directors and actors involved. It would be interesting to see how we could combine these observable features with latent features to produce more accurate predictions.

Finally, in our work, we assume that the data is static. However, in real-world, data is never frozen in time but arrives in the system in a dynamic fashion. For instance, we receive new ratings or a new user/item enters into the system and the matrix to be filled has to grow in the number of rows or columns. We would like to extend our algorithm to this online setting.
Appendix A

Useful Inequalities

Lemma A.0.1 (Chebyshev’s Inequality). Let $X$ (integrable) be a random variable with finite expected value $\mu$ and finite non-zero variance $\sigma^2$. Then for any $k > 0$,

$$\mathbb{P}(|X - \mu| \geq k\sigma) \leq \frac{1}{k^2}$$

Lemma A.0.2 (Hoeffding’s Inequality). If $B \sim \text{Binomial}(n, p)$, then

$$\mathbb{P}(B - np \leq -\epsilon n) \leq \exp(-2\epsilon^2 n)$$
$$\mathbb{P}(B - np \geq \epsilon n) \leq \exp(-2\epsilon^2 n)$$
Bibliography


