LOCALIZING EMBEDDINGS FOR RECOMMENDATION SYSTEMS USING BINARY PAIRWISE COMPARISONS

A Thesis
Presented to
The Academic Faculty

by

Matthew R. O'Shaughnessy

In Partial Fulfillment
of the Requirements for the Degree
Bachelor of Science in Electrical Engineering, Research Option, in the
School of Electrical and Computer Engineering

Georgia Institute of Technology
May 2016

LOCALIZING EMBEDDINGS FOR RECOMMENDATION SYSTEMS USING BINARY PAIRWISE COMPARISONS

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Collaborative filtering systems use data from "user" ratings of various "items" to build a model of user preferences to a group of items. Applications are numerous: in the famous Netflix problem, users represent watchers of the service, and items represent movies or TV shows; in other consumer applications, users represent customers and items represent products. The collaborative filtering problem seeks to use this model to predict items a user may like. Similarly, given an item, the model can be used to predict users that may like the item.

One model used for consumer preference applications is the *ideal point model* of preference, in which users and items are placed in an embedding in $\mathbb{R}^n$ such that the distance between an item and user represents a user's preference for that item. Collaborative filtering seeks to create and use information from such an embedding to inform an estimate of a new user's preferences.

In this work, we consider the case where our *a priori* knowledge of user preferences is only informed by an embedding of users and items and a collection of *binary pairwise comparisons* of the form "user u prefers item i to item j." We show we can accurately localize new users in this embedding using only the collection of binary pairwise comparisons by solving a simple quadratic program. We extend this method to allow us to localize new items by solving a similar optimization problem. For both cases, we consider practical implementation details to enable our algorithm to perform the localization both accurately and computationally efficiently even as the size and dimensionality of the dataset grow large.

Further, we extend this method for localizing new points to an iterative method

for using binary pairwise comparisons to increase the accuracy of a given noisy embedding, and show that we can achieve a significant reduction in mean-squared error given a sufficient number of comparisons, even in high dimensionalities. In the extreme case, where no *a priori* estimate of the embedding is known and our only knowledge of the embedding is given by the list of binary pairwise comparisons, we show that we can accurately recover the entire embedding.

CHAPTER II

BACKGROUND

The collaborative filtering problem is to estimate a user's preference—typically for a product where preference data is easily available, such as movies or music–given prior measurements, or *ratings*, from the user. Using collaborative filtering, connections between the preferences of users and features of items are used to calculate an estimated preference for a specific user. We use the *ideal point model*, which represents user preferences in the same space as item features; for example, in the movie classification scenario, each item (e.g., movie or book) might be represented with a vector of values representing how well it conforms to traits of a certain genre, mood, length, or critical reception. This set of values lives in a highly dimensional space, and the distance between it and another item represents the items' similarity. Psychology research indicates that obtaining the type of pairwise comparisons we consider in this work may be a more accurate method for measuring consumer preference [5].

By considering a known configuration of existing users and items and a set of binary pairwise comparisons involving a new point, we can form a convex optimization problem to allow us to determine a new user or item's position. Stating the problem as a convex optimization problem (similar in form to the optimization problem used in support vector machines [2]) allows us to take advantage of well-studied algorithms for convex optimization. To use the given binary pairwise comparisons to estimate the "true" point that satisfies all comparisons (or, when using noisy *a priori* data, minimizing the ℓ_1-norm of the violations), we first seek to formulate the core optimization problem, which is derived in Chapter III.

In this work, we consider several optimization algorithms. Here, we briefly review the algorithms implemented; the update rules are derived separately for the optimization problem of interest in Section III. The optimization problem may be solved directly with these algorithms, or may be solved using a widely-available software package such as CVX or MATLAB's `quadprog` [4].

The gradient descent algorithm finds the minimum of a cost function by repeatedly calculating the gradient and taking a step in the direction of most negative gradient. By performing this operation iteratively, the algorithm eventually converges on the global (because the cost function is convex) minimum. The step size parameter, which controls the time to convergence and the possible instability (is this the right word?) of the solution must be well-selected for accurate and quick convergence. There are several variants of the gradient descent algorithm that provide subtle performance benefits. *Stochastic gradient descent* reduces the required computation by only calculating an approximation of the gradient at each iteration. *Line search* can provide faster convergence by evaluating several choices of the step size parameter at each iteration, selecting the step size that results in the best improvement in cost.

Another algorithm for solving this optimization problem, the *Newton-Raphson method*, exploits the second derivative of the objective function to automatically select the step size at each iteration. While this requires more computation (recalculation of the Hessian matrix) at each iteration, it may significantly reduce the total number of iterations required for the algorithm to converge to the minimum of the cost function. To calculate the Hessian matrix, the objective function of the optimization problem must be smooth, forcing us to use the logistic function as an approximation of the hinge-loss function in our optimization problem. The *Lagrangian dual*, an equivalent formulation of the optimization problem that may be more computationally efficient to solve in some cases, is derived by using a vector of *Lagrange multipliers* to bring the constraints into the objective function [1].

Characteristics of the dataset (which may be highly application dependent) may have a significant impact on selecting of the most efficient algorithm for solving the optimization problem and finding the true preference point. A significant aspect of this work is in developing “rules of thumb” for selecting an appropriate algorithm and parameters based on the dataset or application.

CHAPTER III

ALGORITHMS

In this section, we present the mathematical formulation of the optimization problems used to localize points in the embedding. Derivations of the strategies and data structures used for efficient implementation are presented in the following chapter.

First, we derive separate optimization problems for localizing users and items. Then, we show that we can express both optimization problems equivalently in a vectorized form using only different constants.

We denote the kth user as $\boldsymbol{x}_k$ and the kth item as $\boldsymbol{q}_k$ (both in $\mathbb{R}^n$). Each binary pairwise comparison of the form "user $\boldsymbol{x}_m$ prefers item $\boldsymbol{q}_i$ to item $\boldsymbol{q}_j$" is denoted $\|\boldsymbol{x}_m - \boldsymbol{q}_i\|_2 \leq \|\boldsymbol{x}_m - \boldsymbol{q}_j\|_2$. Thus, each binary pairwise comparison can be represented by the triple of indices (m, i, j).

3.1 Localizing users

To localize a user in an existing embedding of items and users, we minimize the total magnitude of comparison violations plus an ℓ_2-norm term for regularization. A vector of *slack variables*, $\boldsymbol{\xi}$, is used to allow the optimal value to violate one or more comparisons in the case where the comparisons or embedding is noisy. The vector $\boldsymbol{c} \in \mathbb{R}^K$ contain weights for each comparison that control the trade-off between the contributions of the regularization term and each comparison violation to the objective function. Each binary pairwise comparison is treated as a linear constraint.

The optimization problem is:

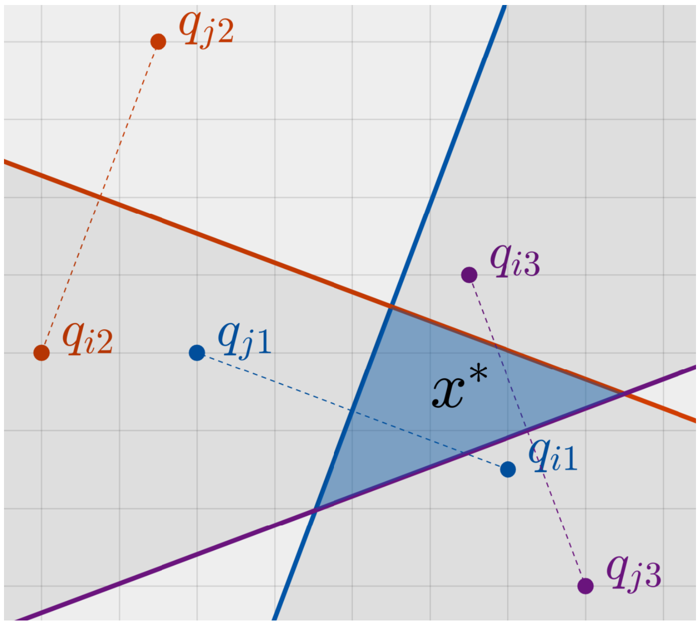


Figure 1: Feasible region induced by three linear constraints. Each linear constraint results from one binary pairwise comparison.

$$\begin{aligned} \underset{\boldsymbol{x},\,\boldsymbol{\xi}}{\text{minimize}} \quad & \tfrac{1}{2}\|\boldsymbol{x}-\boldsymbol{x}_o\|_2^2 + \sum_{k\in\mathcal{T}} c_k\xi_k \\ \text{subject to} \quad & \|\boldsymbol{x}-\boldsymbol{q}_i\|_2^2 \le \|\boldsymbol{x}-\boldsymbol{q}_j\|_2^2 + \xi_k \\ & \xi_k \ge 0 \end{aligned} \tag{1}$$

The regularization term $\frac{1}{2}\|\boldsymbol{x}-\boldsymbol{x}_o\|_2^2$ serves two purposes: (1) it prevents the optimal solution from going to infinity if the set of comparisons does not result in hyperplanes completely bounding the feasible region, resulting in an unbounded solution; and (2) it prevents drastic movement of the estimate of $\boldsymbol{x}$ in cases where a small number of comparisons results in a very large feasible region.

To see that the problem is a standard quadratic program with linear constraints, we equivalently write the optimization problem in standard form as:

$$\begin{aligned}
\underset{\boldsymbol{x},\boldsymbol{\xi}}{\text{minimize}} \quad & \tfrac{1}{2}\|\boldsymbol{x}-\boldsymbol{x}_o\|_2^2 + \sum_{k\in\mathcal{T}} c_k \xi_k \\
\text{subject to} \quad & (\boldsymbol{q}_j-\boldsymbol{q}_i)^T\boldsymbol{x} + \frac{\|\boldsymbol{q}_i\|_2^2-\|\boldsymbol{q}_j\|_2^2}{2} + \xi_k \le 0 \\
& -\xi_k \le 0
\end{aligned} \tag{2}$$

In this form, it is apparent that each constraint resulting from a binary pairwise comparison is linear, defining a hyperplane such that one side of the space split by the hyperplane is the region in $\mathbb{R}^n$ where the "true" user lies. Figure (1) shows a geometric interpretation of the feasible region created by several comparisons.

Finally, we "vectorize" the optimization problem by enumerating entries of $(\boldsymbol{q}_j - \boldsymbol{q}_i)$ as the columns of matrix $\boldsymbol{A}$ and the scalar entries $\frac{1}{2}(\|\boldsymbol{q}_i\|_2^2 - \|\boldsymbol{q}_j\|_2^2)$ as entries of the column vector $\boldsymbol{b}$. Thus, the optimization problem takes its final form as:

$$\begin{aligned}
\underset{\boldsymbol{x},\boldsymbol{\xi}}{\text{minimize}} \quad & \tfrac{1}{2}\|\boldsymbol{x}-\boldsymbol{x}_o\|_2^2 + \boldsymbol{c}^T\boldsymbol{\xi} \\
\text{subject to} \quad & A\boldsymbol{x} + \boldsymbol{b} + \boldsymbol{\xi} \le 0 \\
& -\boldsymbol{\xi} \le 0
\end{aligned} \tag{3}$$

where the inequality sign in both constraints is applied element-wise.

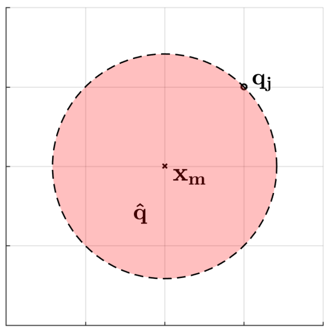
3.2 Localizing items

3.2.1 Deriving the optimization problem

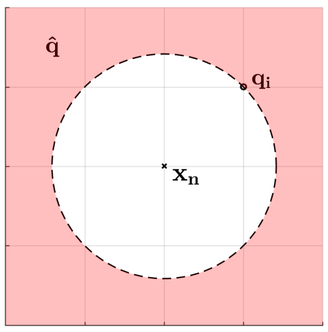
The optimization problem for localizing items is set up in the same way, minimizing the sum of comparison violations plus a regularization term and using a vector of slack variables to allow the solution to violate some comparisons:

$$
\begin{aligned}
\underset{\hat{\boldsymbol{q}},\boldsymbol{\xi}}{\text{minimize}} \quad & \frac{1}{2}\|\hat{\boldsymbol{q}} - \boldsymbol{q}_o\|_2^2 + \sum_{k\in\mathcal{T}} c_k\xi_k \\
\text{subject to} \quad & \|\boldsymbol{x}_m - \hat{\boldsymbol{q}}\|_2^2 \leq \|\boldsymbol{x}_m - \boldsymbol{q}_j\|_2^2 + \xi_k \\
& \|\boldsymbol{x}_n - \boldsymbol{q}_i\|_2^2 \leq \|\boldsymbol{x}_n - \hat{\boldsymbol{q}}\|_2^2 + \xi_k \\
& \xi_k \geq 0
\end{aligned}
\tag{4}
$$

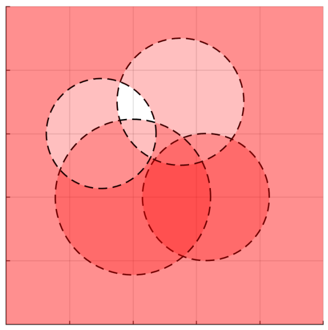
Here, the comparisons creating constraints are divided into two groups: those for which the item to localize $\hat{\boldsymbol{q}}$ represents the more preferred item ($\hat{\boldsymbol{q}} = \boldsymbol{q}_i$), and those for which the item to localize $\hat{\boldsymbol{q}}$ represents the less preferred item ($\hat{\boldsymbol{q}} = \boldsymbol{q}_j$).



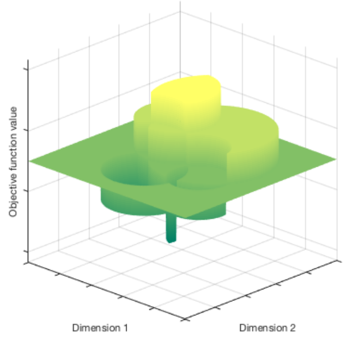
(a) Constraint resulting in convex feasible region



(b) Constraint resulting in non-convex feasible region



(c) Feasible region resulting from multiple constraints



(d) Non-convex objective function resulting from many constraints

Figure 2: Objective function in $\mathbb{R}^2$ for (a) comparisons where the item to localize is the "preferred" item ($\hat{\boldsymbol{q}} = \boldsymbol{q_i}$) and (b) comparisons where the item to localize is the "less preferred" item ($\hat{\boldsymbol{q}} = \boldsymbol{q_j}$). The combination of both types of constraints results in a highly non-convex feasible region, as shown in (c-d).

The feasible region resulting from these two types of comparison is illustrated in $\mathbb{R}^2$ in Figure 2. Comparisons where the item to be localized is the "more preferred" item ($\hat{\boldsymbol{q}} = \boldsymbol{q}_i$) can be interpreted as a upper bound on the distance from the user; therefore, they create a convex feasible region as in subfigure (a). In contrast, comparisons where the item to be localized is the "less preferred" item ($\hat{\boldsymbol{q}} = \boldsymbol{q}_j$) can be interpreted as an lower bound on the distance from the user; therefore, they create a non-convex feasible region as in subfigure (b). When multiple comparisons of these two types are combined (subfigure (c)), the resulting objective function is non-convex (subfigure (d)).

3.2.2 Making the item localization problem convex

To make this problem convex, we can strategically *combine* comparisons, creating a single linear constraint by adding two comparisons. Given two comparisons:

$$\begin{cases} \|\boldsymbol{x}_1 - \boldsymbol{q}_{1,i}\|_2^2 \leq \|\boldsymbol{x}_1 - \boldsymbol{q}_{1,j}\|_2^2 \\ \|\boldsymbol{x}_2 - \boldsymbol{q}_{2,i}\|_2^2 \leq \|\boldsymbol{x}_2 - \boldsymbol{q}_{2,j}\|_2^2 \end{cases} \tag{5}$$

We can add them if $\hat{\boldsymbol{q}} = \boldsymbol{q}_{1,j} = \boldsymbol{q}_{2,i}$ or $\hat{\boldsymbol{q}} = \boldsymbol{q}_{1,i} = \boldsymbol{q}_{2,j}$, leading to two cases:

$$\begin{cases} \quad \|\boldsymbol{x}_1 - \hat{\boldsymbol{q}}\|_2^2 \leq \overbrace{\|\boldsymbol{x}_1 - \boldsymbol{q}_{1,j}\|_2^2}^{c} \\ + \quad \underbrace{\|\boldsymbol{x}_2 - \boldsymbol{q}_{2,i}\|_2^2}_{d} \leq \|\boldsymbol{x}_2 - \hat{\boldsymbol{q}}\|_2^2 \end{cases} \tag{6}$$

$$\begin{cases} \quad \overbrace{\|\boldsymbol{x}_1 - \boldsymbol{q}_{1,i}\|_2^2}^{c} \leq \|\boldsymbol{x}_1 - \hat{\boldsymbol{q}}\|_2^2 \\ + \quad \|\boldsymbol{x}_2 - \hat{\boldsymbol{q}}\|_2^2 \leq \underbrace{\|\boldsymbol{x}_2 - \boldsymbol{q}_{2,j}\|_2^2}_{d} \end{cases} \tag{7}$$

For each inequality, the side not containing the item to localize is a constant, which we denote c and d. The second case can be written as the first with the order of the comparisons switched; therefore, henceforth we consider only the first case. Expanding, we note that combining the comparisons results in the cancellation of the quadratic term $\|\hat{\boldsymbol{q}}\|_2^2$, simplifying the computationally intractable quadratic constraint into a linear one. Simplifying yields:

$$-2\boldsymbol{x}_1^T\hat{\boldsymbol{q}} + \|\boldsymbol{x}_1\|_2^2 + d \leq -2\boldsymbol{x}_2\hat{\boldsymbol{q}} + \|\boldsymbol{x}_2\|_2^2 + c$$

$$\Rightarrow \hat{\boldsymbol{q}}^T(\boldsymbol{x}_2 - \boldsymbol{x}_1) < \frac{1}{2}\left(c - d + \|\boldsymbol{x}_2\|_2^2 - \|\boldsymbol{x}_1\|_2^2\right) \tag{8}$$

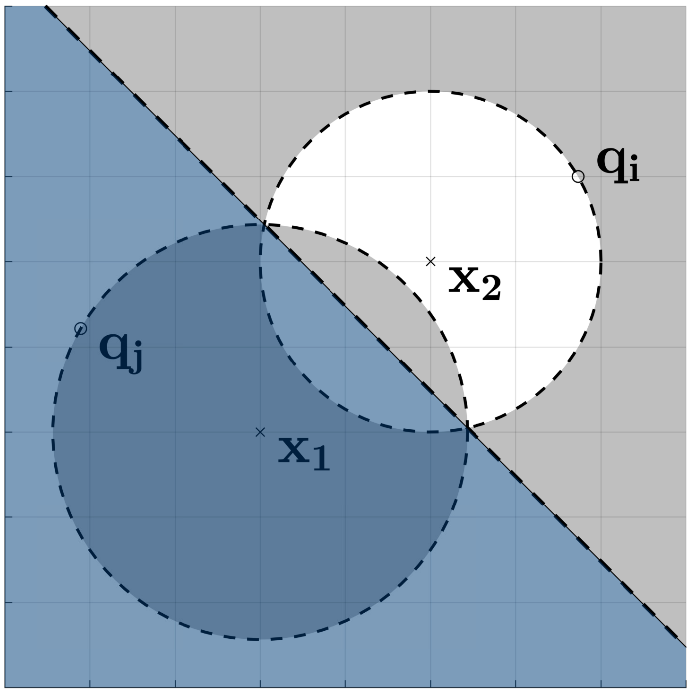


Figure 3: Linear constraint generated by the combination of two quadratic constraints. Gray shading represents the feasible regions resulting from each of the two binary pairwise comparisons involving the item to localize; blue shading represents the feasible region resulting from the resulting linear combination.

This form yields additional geometric intuition of the linear constraint created. As with constraints in the user localization problem in Equation 2, the constraints define hyperplanes in $\mathbb{R}^n$. Figure 3 shows a geometric interpretation of the two types of quadratic constraints resulting in a non-convex optimization problem and the linear constraint induced by a combination of the two quadratic constraints.

More discussion of methods for choosing the best comparisons to combine is in the following chapter.

Finally, we "vectorize" the problem into the same format as the optimization problem for localizing users (2), by enumerating entries of $(\boldsymbol{x}_2 - \boldsymbol{x}_1)$ as the rows of

A and the scalar entries $\frac{1}{2}\left(c - d + \|\boldsymbol{x}_2\|_2^2 - \|\boldsymbol{x}_1\|_2^2\right)$ as the entries of column vector $\boldsymbol{b}$. Thus, the optimization problem for localizing items takes its final form as:

$$\begin{aligned} \underset{\hat{\boldsymbol{q}},\boldsymbol{\xi}}{\text{minimize}} \quad & \tfrac{1}{2}\|\hat{\boldsymbol{q}} - \boldsymbol{q}_o\|_2^2 + \boldsymbol{c}^T\boldsymbol{\xi} \\ \text{subject to} \quad & \boldsymbol{A}\hat{\boldsymbol{q}} + \boldsymbol{b} + \boldsymbol{\xi} \leq \boldsymbol{0} \\ & -\boldsymbol{\xi} \leq \boldsymbol{0} \end{aligned} \tag{9}$$

Note that this is the same form as the optimization problem formulated for localizing users. Therefore, although the most accurate and fastest method for solving the optimization problem for localizing users and items may differ, we can use identical implementations of each method by simply using a different $\boldsymbol{A}$ and $\boldsymbol{b}$.

3.3 Summary: a vectorized form for localizing users and items

Given an embedding of items and users in $\mathbb{R}^n$, we can localize any item or user using only comparisons of the form $\|\boldsymbol{x}_u - \boldsymbol{q}_i\|_2 \leq \|\boldsymbol{x}_u - \boldsymbol{q}_j\|_2$ by solving the optimization problem:

$$\begin{aligned} \underset{\hat{\boldsymbol{p}},\boldsymbol{\xi}}{\text{minimize}} \quad & \tfrac{1}{2}\|\hat{\boldsymbol{p}} - \boldsymbol{p}_o\|_2^2 + \boldsymbol{c}^T\boldsymbol{\xi} \\ \text{subject to} \quad & \boldsymbol{A}\hat{\boldsymbol{p}} + \boldsymbol{b} + \boldsymbol{\xi} \leq \boldsymbol{0} \\ & -\boldsymbol{\xi} \leq \boldsymbol{0} \end{aligned} \tag{10}$$

where $\boldsymbol{p}$ is the point (item or user) to localize, $\hat{\boldsymbol{p}}$ is the localized point, $\boldsymbol{p}_o$ is the previous estimate of $\boldsymbol{p}$, and $\boldsymbol{c}$ is the vector of constants that controls the relative influence of each comparison. $\boldsymbol{A}$ and $\boldsymbol{b}$ are defined by the comparisons involving point being located. Here, A_k denotes the kth column of the matrix $\boldsymbol{A} \in \mathbb{R}^{K\times d}$ and b_k denotes the kth element of the column vector $\boldsymbol{b}$.

	User Localization	Item Localization
A_k	$(\boldsymbol{q}_j - \boldsymbol{q}_i)$	$(\boldsymbol{x}_1 - \boldsymbol{x}_2)$
b_k	$\frac{1}{2}(\\|\boldsymbol{q}_i\\|_2^2 - \\|\boldsymbol{q}_j\\|_2^2)$	$+\frac{1}{2}\left(c - d + \\|\boldsymbol{x}_2\\|_2^2 - \\|\boldsymbol{x}_1\\|_2^2\right)$ where $c = \\|\boldsymbol{x}_2 - \boldsymbol{q}_{2,j}\\|_2^2$, $d = \\|\boldsymbol{x}_2 - \boldsymbol{q}_{2,i}\\|_2^2$

3.4 The primal and dual optimization problems

3.4.1 The primal

The Lagrangian function resulting from Equation 10, which brings the constraints into the objective function, is:

$$L(\hat{\boldsymbol{p}}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{1}{2}(\hat{\boldsymbol{p}} - \boldsymbol{p}_o)^T(\hat{\boldsymbol{p}} - \boldsymbol{p}_o) + \sum_k c_k \xi_k + \sum_k \alpha_k (b_k - A_k^T \hat{\boldsymbol{p}} - \xi_k) - \sum_k \beta_k \xi_k \quad (11)$$

where $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are vectors of the Lagrange multipliers.

The primal function is then:

$$L_P(\hat{\boldsymbol{p}}) = \max_{\boldsymbol{\alpha}, \boldsymbol{\beta} : \alpha_k \geq 0} L(\hat{\boldsymbol{p}}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) \quad (12)$$

where each α_i is constrained to be positive to ensure each inequality constraint in Equation 10 is satisfied.

The primal optimization problem, which encompasses the original optimization problem in (10) but is unconstrained, is:

$$\min_{\hat{\boldsymbol{p}}, \boldsymbol{\xi}} \max_{\boldsymbol{\alpha}, \boldsymbol{\beta} : \alpha_i \geq 0} L(\hat{\boldsymbol{p}}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) \quad (13)$$

3.4.2 The dual and the dual optimization problem

The Lagrangian dual is:

$$L_D(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \min_{\hat{\boldsymbol{p}}, \boldsymbol{\xi}} L(\hat{\boldsymbol{p}}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) \quad (14)$$

And the dual optimization problem is:

$$\max_{\boldsymbol{\alpha}, \boldsymbol{\beta} : \alpha_k, \beta_k \geq 0} L_D(\boldsymbol{\alpha}, \boldsymbol{\beta}) \quad (15)$$

The primal and dual optimization problems have zero duality gap (share the same solution) when the KKT conditions are met. In Equation 15, $\alpha_k, \beta_k \geq 0$ holds because of the nonnegative KKT condition. We use the remaining KKT conditions to derive $L_D(\boldsymbol{\alpha}, \boldsymbol{\beta})$, starting with the condition that the gradient of the Lagrangian function with respect to each of the primal variables ($\hat{\boldsymbol{p}}$ and ξ_k) must be zero:

$$\nabla_{\hat{\boldsymbol{p}}} L(\boldsymbol{x}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \hat{\boldsymbol{p}} - \boldsymbol{p}_o - \sum_k \alpha_k \boldsymbol{a}_k = 0 \tag{16}$$

$$\nabla_{\xi_k} L(\boldsymbol{x}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = c_k - \alpha_k - \beta_k = 0 \tag{17}$$

Thus, $\hat{\boldsymbol{p}} = \sum_k \beta_k + \boldsymbol{p}_o$. Plugging this in to (11), we obtain:

$$\begin{aligned} L(\hat{\boldsymbol{p}}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{1}{2}\left(\sum_k \alpha_k \boldsymbol{a}_k\right)^T \left(\sum_k \alpha_k \boldsymbol{a}_k\right) + \sum_k c_k \xi_k \\ - \sum_k \beta_k \xi_k + \sum_k \alpha_k \left(b_k - \boldsymbol{a}_k^T \left(\sum_{k'} \alpha_{k'} \boldsymbol{a}_{k'} + \boldsymbol{p}_o\right) - \xi_k\right) \end{aligned} \tag{18}$$

$$= -\frac{1}{2} \sum_{k,k'} \alpha_k \alpha_{k'} \boldsymbol{a}_k^T \boldsymbol{a}_{k'} + \sum_k \alpha_k \left(b_k - \boldsymbol{a}_k^T \boldsymbol{p}_o\right) \tag{19}$$

While β_k has been eliminated from the dual function, to enforce (17) we must add the constraint that $0 \leq \alpha_k \leq c_k$. Adding this as the constraint, we obtain the dual optimization problem:

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & -\frac{1}{2} \sum_{k,k'} \alpha_k \alpha_{k'} \boldsymbol{a}_k^T \boldsymbol{a}_{k'} + \sum_k \alpha_k \left(b_k - \boldsymbol{a}_k^T \boldsymbol{p}_o\right) \\ \text{subject to} \quad & 0 \leq \alpha_k \leq c_k \end{aligned} \tag{20}$$

Defining the matrix $\boldsymbol{G}$ with entries $G_{ij} = A_i^T A_j$, we can equivalently write:

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & -\tfrac{1}{2} \boldsymbol{\alpha}^T \boldsymbol{G} \boldsymbol{\alpha} + \boldsymbol{b}^T \boldsymbol{\alpha} \\ \text{subject to} \quad & 0 \leq \alpha_k \leq c_k \end{aligned} \tag{21}$$

3.5 Iteratively localizing the entire embedding

Using the localization procedures derived above, we can determine the location of a new point in an embedding of items and users given only a list of binary pairwise comparisons. However, our localization procedures can also be used to increase the accuracy of an existing item or user in an embedding. When presented with a noisy embedding, we can iterate through each point and use the localization procedure to improve its accuracy. Because the localization procedure for each point depends on the accuracy of the other points in the embedding, the update in each step can dramatically improve the accuracy of updates in future iterations.

```
Require: noisy embedding, list of comparisons
  for $\ell = 1 \ldots \text{numIters}$ do
      for $i = 1 \ldots \text{nUsers}$ do
          $\boldsymbol{x}_i \leftarrow \text{localize}(\boldsymbol{x}_i)$ using comparisons involving $\boldsymbol{x}_i$
      end for
      for $i = 1 \ldots \text{nItems}$ do
          combine comparisons involving $\boldsymbol{q}_i$
          $\boldsymbol{q}_i \leftarrow \text{localize}(\boldsymbol{q}_i)$ using combined comparisons
      end for
  end for
```

CHAPTER IV

IMPLEMENTATION

In this the previous chapter, we derived an optimization problem for localizing points in a noisy embedding. For localizing items, we demonstrated a method of combining comparisons to make the problem convex. In this chapter, we present efficient methods for calculating the solution to the optimization problem, combining comparisons, and iteratively increasing the accuracy of a noisy embedding.

4.1 Optimization methods

In this section, we derive several well-known optimization methods for solving the localization problem for items and users. In the experiments section, each method is compared in terms of accuracy and speed of convergence for both item and user localization.

4.1.1 Gradient descent using the primal

To optimize the primal, we use the *hinge loss function*, $f(x) = \max(0, x)$, which penalizes positive values proportionally to their magnitude but does not penalize negative values. In the context of this problem, positive values of $b_k - A_k^T \boldsymbol{p}$ indicate comparison k is violated, while a negative value indicates that comparison k was satisfied.

$$\min_{\boldsymbol{p}} \ \tfrac{1}{2}\|\boldsymbol{p} - \boldsymbol{p}_o\|_2^2 + \sum_k c_k \max\left(0, b_k - A_k^T \boldsymbol{p}\right) \tag{22}$$

In gradient descent, we iteratively step towards the minimum of the objective function. Each step is taken in the direction of the objective function's negative gradient evaluated at the current estimate.

Each iteration takes the form:

$$\boldsymbol{p}^{(\ell+1)} = \boldsymbol{p}^{(\ell)} - \eta_\ell \nabla_\ell \tag{23}$$

The gradient is calculated as:

$$\nabla_\ell = \boldsymbol{p}^{(\ell)} - \sum_{k:\boldsymbol{a}_k^T \boldsymbol{p}^{(\ell)} < b_k} c_k \boldsymbol{a}_k \tag{24}$$

where A_k is the kth column of A. The ks that satisfy $A_k^T \boldsymbol{p}^{(\ell)} < b_k$ correspond to the comparisons that are violated.

The step size η_ℓ is a function of the iteration number; in the experiments presented in chapter 5 we use $\eta_\ell = \eta/\ell$.

4.1.2 Stochastic gradient descent using the primal

Rather than use every k at each iteration, in stochastic gradient descent we pick certain ks to update, increasing the speed of each iteration. The ks to update in each iteration can be chosen arbitrarily or according to some rule.

4.1.3 Coordinate descent using the dual

In coordinate descent, we iterate through the dual variables α_k to optimize the dual optimization problem. At each iteration, we solve a simpler version of the optimization problem by minimizing in only one dimension, allowing us to calculate the optimal step size.

The value of the objective function after taking a step of size d in dimension k is

$$-\frac{1}{2}(\boldsymbol{\alpha} + d\boldsymbol{e}_k)^T G(\boldsymbol{\alpha} + d\boldsymbol{e}_k) + \boldsymbol{b}^T(\boldsymbol{\alpha} + d\boldsymbol{e}_k) \tag{25}$$

At each step, maximizing the dual objective function with the constraint $0 \leq \alpha_k + d \leq c_k$ gives us the optimal step size d. Thus, each iteration consists of the updates

$$
\begin{aligned}
d^{(\ell)} &= \frac{b_k - \boldsymbol{a}_k^T \boldsymbol{p}^{(\ell)}}{\boldsymbol{a}_k^T \boldsymbol{a}_k} \\
\alpha_k^{(\ell+1)} &= \min(\max(\alpha_k^{(\ell)} + d^{(\ell)}, 0), c_k) \\
\boldsymbol{p}^{(\ell+1)} &= \boldsymbol{p}^{(\ell)} + (\alpha_k^{(\ell+1)} - \alpha_k^{(\ell)})\boldsymbol{a}_k
\end{aligned} \tag{26}
$$

4.1.4 Newton-Raphson method

The Newton-Raphson method automatically selects the step size parameter in the gradient descent update rule of Equation 23 by using the second derivative of the objective function. To take the second derivative, we must ensure that the objective function is differentiable. Because we constrain $\boldsymbol{\xi} \geq 0$, the ℓ_1 penalization term acts as the *hinge-loss* function $\ell(t) = \max(0, t)$. To make this term differentiable at $t = 0$ and allow us to apply the Newton-Raphson method, we use the smooth approximation of the hinge-loss function $g(t) = \log(1 + e^t)$.

Applying this to the objective function in the vectorized optimization problem of Eq. 10, we obtain

$$
\begin{aligned}
\underset{\hat{\boldsymbol{p}}, \boldsymbol{\xi}}{\text{minimize}} \quad & \tfrac{1}{2}\|\hat{\boldsymbol{p}} - \boldsymbol{p}_o\|_2^2 + \log(1 + \exp(b_k - A_k \boldsymbol{x})) \\
\text{subject to} \quad & \boldsymbol{A}\hat{\boldsymbol{p}} + \boldsymbol{b} + \boldsymbol{\xi} \leq \boldsymbol{0} \\
& -\boldsymbol{\xi} \leq \boldsymbol{0}
\end{aligned} \tag{27}
$$

where entries of the vector $\boldsymbol{b}$ and columns of the matrix $\boldsymbol{A}$ are as defined in Table 1.

The Hessian matrix is:

$$
\boldsymbol{H}(\boldsymbol{x}) = \boldsymbol{I} + \sum_k \boldsymbol{a}_k \boldsymbol{a}_k^T g(b_k - \boldsymbol{a}_k^T \boldsymbol{x})(1 - g(b_k - \boldsymbol{a}_k^T \boldsymbol{x})) \tag{28}
$$

The optimal step is given by $\boldsymbol{H}(\boldsymbol{x}^{(\ell-1)})^{-1}\nabla(\boldsymbol{x}^{(\ell-1)})$, so each iteration for the Newton-Raphson method is the update:

$$
\boldsymbol{x}^{(\ell)} = \boldsymbol{x}^{(\ell-1)} - \boldsymbol{H}(\boldsymbol{x}^{(\ell-1)})^{-1}\nabla(\boldsymbol{x}^{(\ell-1)}) \tag{29}
$$

4.2 Choosing comparisons to combine for item localization

As discussed in Section II, we can combine two comparisons, $\|\boldsymbol{x}_1-\boldsymbol{q}_{1,i}\|_2^2 \leq \|\boldsymbol{x}_1-\boldsymbol{q}_{1,j}\|_2^2$ and $\|\boldsymbol{x}_2 - \boldsymbol{q}_{2,i}\|_2^2 \leq \|\boldsymbol{x}_2 - \boldsymbol{q}_{2,j}\|_2^2$ to form a valid constraint if $\hat{\boldsymbol{q}} = \boldsymbol{q}_{1,j} = \boldsymbol{q}_{2,i}$ or if $\hat{\boldsymbol{q}} = \boldsymbol{q}_{1,i} = \boldsymbol{q}_{2,j}$. Using synthetically generated data, application of this rule typically gives us many times more convex constraints from combined comparisons than we had quadratic constraints from the original comparisons. However, most of these combined comparisons result in hyperplanes that are very far from the feasible region. Thus, the majority of combined comparisons do not contribute new information and only increase the computational cost of solving the optimization problem, suggesting that we can potentially dramatically reduce the number of combinations of binary pairwise comparisons we require. To reduce the computational cost, we select only a random subset of combinations to use as constraints in the optimization problem; this simplification had very little impact on localization accuracy in our experiments.

CHAPTER V

EXPERIMENTS AND RESULTS

5.1 Localizing a new user

First, we demonstrate the first typical application of localizing a new user. In these experiments, we have *a priori* knowledge of an existing embedding of users and items, and use our list of binary pairwise comparisons to estimate the preferences (modeled by a point in the embedding) of a new user.

In each experiment, a "ground truth" embedding of many items and users was generated. Comparisons were generated by randomly selecting a user $\boldsymbol{x_k}$ and two items $\boldsymbol{q_1}$ and $\boldsymbol{q_2}$. Providing $\boldsymbol{q_1} \neq \boldsymbol{q_2}$. $\boldsymbol{q_1}$ and $\boldsymbol{q_2}$ were then selected such that $\|\boldsymbol{x} - \boldsymbol{q_i}\|_2 < \|\boldsymbol{x} - \boldsymbol{q_j}\|_2$, and the triple of indices (k, i, j) representing the comparison was added to the list of comparisons.

In some experiments, noise was added to the comparisons. We model two types of noise in the comparisons:

1. *Noise in selection of preferred item.* We represent this type of error by adding a noise term ϵ when selecting which item is the preferred item $\boldsymbol{q_i}$. Thus, $\boldsymbol{q_1}$ is selected as $\boldsymbol{q_i}$ if $\|\boldsymbol{x} - \boldsymbol{q_i}\|_2 + \epsilon < \|\boldsymbol{x} - \boldsymbol{q_j}\|_2$, where ϵ is a random variable. This type of noise is more likely to "flip" a comparison where $\|\boldsymbol{x} - \boldsymbol{q_i}\|_2$ and $\|\boldsymbol{x} - \boldsymbol{q_j}\|_2$ have similar magnitudes.

2. *Noise in storage of comparison.* We represent this type of error by flipping the more preferred item $\boldsymbol{q_i}$ and the less preferred item $\boldsymbol{q_j}$ with probability P_{err}. This type of noise is equally likely to "flip" any comparison, regardless of the relative values of $\|\boldsymbol{x} - \boldsymbol{q_i}\|_2$ and $\|\boldsymbol{x} - \boldsymbol{q_j}\|_2$.

We use two metrics for the accuracy of the reconstruction. First, we use the ℓ_2 *reconstruction error*: the distance between the localized user and the ground-truth point comparisons were generated from. This gives us an exact metric for the localization error, but is highly dependent on the number of comparisons involving the user localized; with a small amount of comparisons involving the user to localize, there is not enough information to accurately reconstruct the user. A second metric, *number of comparisons violated*, gives us a sense of how accurately the user is localized based only on the information we have available. In the following experiments, we use both metrics, representing the number of comparisons violated as a percentage of the total comparisons involving the user to localize we have available.

5.1.1 Localization accuracy with varying number of comparisons

In the first experiment, we demonstrate the accuracy with which we can localize a new user based on the number of binary pairwise comparisons we have available.

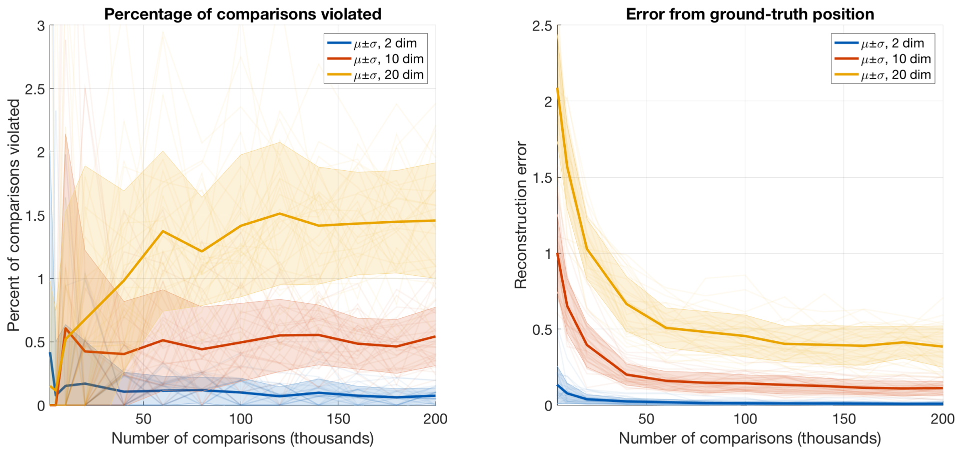


Figure 4: New user recovery error measured in terms of percent of binary comparisons violated and ℓ_2 reconstruction error as a function of the total number of comparisons available for the system. Percent binary comparisons violated refers to the number of comparisons involving the new user. For both metrics, the comparisons were generated uniformly from 100 items and 100 users, and the "ground truth" embedding was generated on $[-1, 1]$ in each dimension.

As shown in Figure 4, a small number of comparisons involving the new user results in a high reconstruction error, while after a certain point a large number of comparisons does not result in significant accuracy improvements. Note that in these experiments, the comparisons are generated randomly: a significantly smaller number of comparisons would be necessary to achieve similar performance if they were selected intelligently.

The dimensionality of the space has a large impact on the ℓ_2 recovery error. This is expected; in the space spanning $[-1, 1]$ in each of d dimensions, the maximum possible error is $\sqrt{d2^d}$. Figure 4 does not normalize for the dimensionality.

5.1.2 Localization accuracy with a noisy *a priori* embedding

In this section, we show that we can accurately localize a new user even if the embedding we are given *a priori* is noisy. For this experiment, the comparisons were generated from the noise-free ground-truth embedding, but our knowledge of the item locations the binary pairwise comparisons refer to are contaminated with noise of standard deviation σ in each dimension.

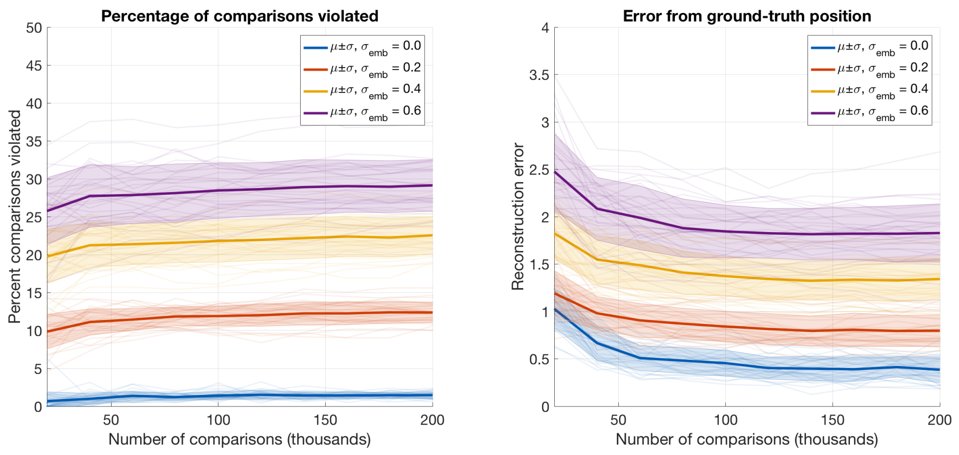


Figure 5: New user recovery error measured in terms of percent of comparisons violated and ℓ_2 recovery error for a new user localized using an embedding contaminated with four different levels of noise. For both metrics, the comparisons were generated uniformly from 100 users and 100 items, and the "ground truth" embedding was generated on $[-1, 1]$ in $d = 20$ dimensions.

Figure 5 shows that even small amounts of noise have a significant detrimental effect on the accuracy with which we can localize a new user. Because the noise added to the embedding may make many of the comparisons conflict with each other, significant error is observed in both the portion of comparisons violated and the ℓ_2 reconstruction metric.

This result motivates our work to improve the accuracy of a noisy embedding by iteratively localizing every point in an embedding. Results that improve on the performance in this experiment are shown in subsequent sections.

5.2 Localizing a new item

Next, we test our method of combining binary pairwise comparisons to create linear constraints for localizing a new item given an existing embedding of users and items and a list of binary pairwise comparisons involving the new item. Because the binary pairwise comparisons result in *quadratic* constraints when localizing an item, we use the convex relaxation described in previous chapters to combine them to form linear constraints.

5.2.1 Localization accuracy with varying number of comparisons

In this experiment, we show that by using our method for combining binary pairwise constraints, we can localize a new item into an existing embedding with similar accuracy as the new user localization problem.

Figure 6 parallels Figure 4 and demonstrates that we can accurately localize an item with approximately the same accuracy as we can localize a new user.

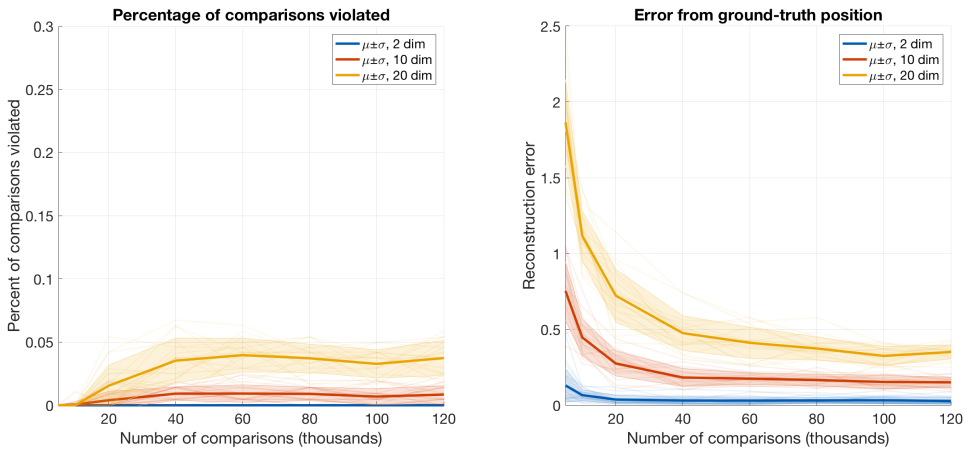


Figure 6: New item recovery error measured in terms of percent of binary comparisons violated and ℓ_2 reconstruction error as a function of the total number of comparisons available for the system. Percent binary comparisons violated refers to the number of comparisons involving the new user. For both metrics, the comparisons were generated uniformly from 100 items and 100 users, and the "ground truth" embedding was generated on $[-1, 1]$ in each dimension. Note that the scale on the y-axis of the first plot is extremely small!

5.2.2 Localization accuracy with a noisy *a priori embedding*

Here, we show that we can also accurately localize a new item even if the embedding we are given *a priori* contains noise. As in the similar experiment for localizing a new user given a noisy embedding, the comparisons were generated from the noise-free ground truth embedding, but our knowledge of the users and items they refer to is contaminated with noise of standard deviation σ in each dimension.

We conclude that using our method for combining comparisons, we can localize new items with the same method we use for localizing new users, with approximately the same accuracy. This result holds for both a noise-free and noisy *a priori* embedding of users and items.

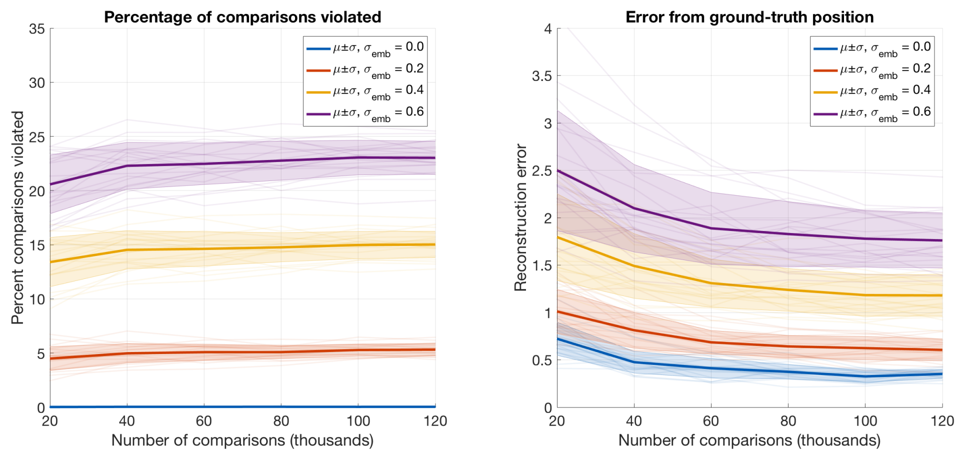


Figure 7: New item recovery error measured in terms of percent of comparisons violated and ℓ_2 recovery error for a new user localized using an embedding contaminated with four different levels of noise. For both metrics, the comparisons were generated uniformly from 100 users and 100 items, and the "ground truth" embedding was generated on $[-1, 1]$ in $d = 20$ dimensions.

5.3 Improving the accuracy of a noisy embedding

In many applications, we may have a rough estimate of the embedding of users and items. In the previous sections, we have shown that we can localize a new user or item to extremely high accuracy if our embedding is not noisy and we have a sufficient number of comparisons involving the new user. Further, we have shown that we can localize a new user or item even if the *a priori* embedding is contaminated with noise.

In this section, we show that we can improve the performance in the noisy case by iteratively localizing each user and item in the embedding. Not only does this result in significantly improved performance when localizing a new user or item, it also results in a much more accurate embedding, up to a scale, shift, and rotation.

5.3.1 Improving the accuracy of a noise-contaminated embedding

In this experiment, we generate an *a priori* embedding of users and items, which we call the ground truth embedding. Then, we contaminate every point in the embedding with additive Gaussian noise of standard deviation σ.

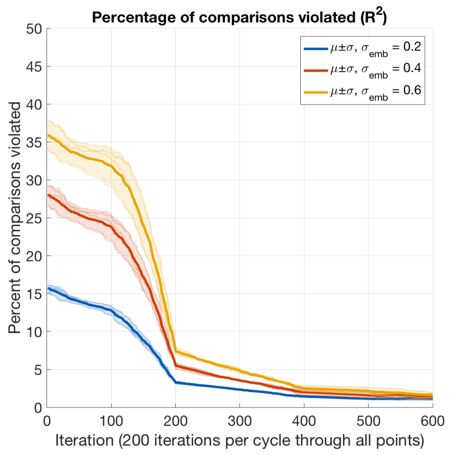
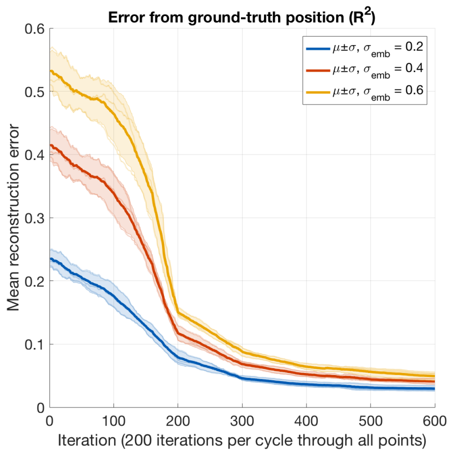


Figure 8: Noisy embedding recovery error measured in terms of percent of all comparisons violated and mean ℓ_2 recovery error for an embedding contaminated with three different levels of noise. For both metrics, the comparisons were generated uniformly from 100 users and 100 items, and the "ground truth" embedding was generated on $[-1, 1]$ in $d = 2$ dimensions. Three trials were performed at each noise level.

Once again, we measure localization accuracy with both the total number of comparisons violated and the mean ℓ_2 reconstruction error of each point in the embedding. Here, the comparisons violated are expressed as the portion of *all* binary pairwise comparisons we are given, not only the comparisons involving the user or item we are localizing at that iteration.

As shown in Figure 8 (in $\mathbb{R}^2$), even relatively high levels of noise converge to a small mean-square error within just a few cycles through the data.

In this experiment, there are 100 users and 100 items, so each cycle consists of 200 iterations. Interestingly, while the first iteration through each user results in significant improvement, the first iteration through each item shows the most dramatic improvement. After the first cycle, improvement is more gradual.

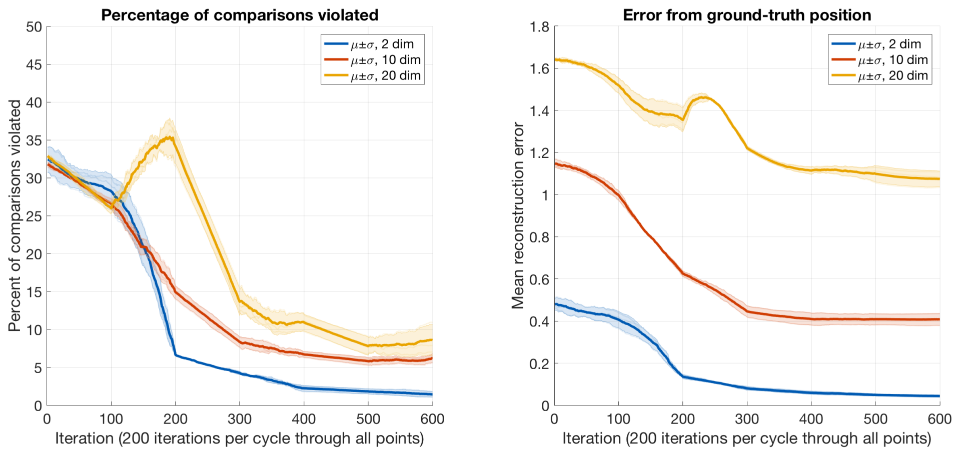


Figure 9: Noisy embedding recovery error measured in terms of percent of all comparisons violated and mean ℓ_2 recovery error for an embedding contaminated with three different levels of noise. For both metrics, the comparisons were generated uniformly from 100 users and 100 items, and the "ground truth" embedding was generated on $[-1, 1]$ in d dimensions. Three trials were performed for each dimensionality.

5.3.2 Improving the accuracy of a noise-contaminated embedding in higher dimensionalities

In this experiment, we extend the results of the previous section, which showed we could significantly improve the accuracy of a noisy embedding in $\mathbb{R}^2$, to higher dimensionalities.

Figure 9 demonstrates that our algorithm also improves the accuracy of a noisy embedding in higher dimensionalities, although higher dimensionalities suffer a performance penalty in both the portion of comparisons violated and the ℓ_2 reconstruction error.

5.4 Generating an embedding with no a priori *knowledge*

In the previous section, we demonstrated good results localizing a noisy embedding even when it was corrupted by enough noise to make the original embedding violate 35% of the binary pairwise comparisons. In this section, we show that even more impressively, we can obtain similar results when we have no *a priori* knowledge of the embedding, generating an embedding using only the list of binary pairwise comparisons.

In some applications, we may have no knowledge of the *a priori* embedding and be presented with only a list of binary pairwise comparisons. Here, we show that our algorithm can construct a embedding that satisfies most or all comparisons with no *a priori* knowledge of the embedding.

5.4.1 Generating an embedding from only a list of comparisons

In this experiment, we again create a "ground truth" embedding of users and items distributed uniformly randomly on $[-1, 1]$ in each dimension. Using this, we generate a list of binary pairwise comparisons, then initialize the reconstructed embedding to a random set of points and use only the list of comparisons to attempt to recreate the embedding. As in the previous experiments, we measure the accuracy of our reconstruction with a shift, scale, and rotation invariant mean error between each point and its actual value and the number of comparisons the reconstructed embedding violates.

Figure 10 shows that even with a random initial embedding (approximately 50% of comparisons violated), we can generate an embedding with only a modest error. While the mean ℓ_2 recovery error in Figure 10(b) shows increasing error as the dimensionality increases, the error expressed in terms of the number of comparisons violated does not continue to increase as the dimensionality grows large, suggesting that the only error increasing the dimensionality results in is due to the increasing vastness of Euclidean dimensional space. This result is particularly encouraging because many applications require highly dimensional embeddings to achieve good results.

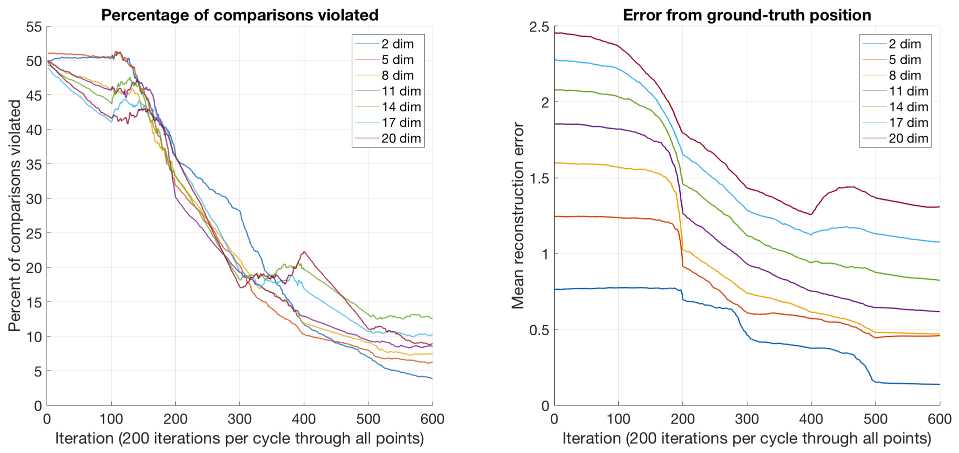


Figure 10: Recovery error for a generated embedding measured in terms of percent of all comparisons violated and mean ℓ_2 recovery error, using the Procrustes distance to ignore the effects of a shift, scale, or rotation from the initial embedding. For both metrics, the comparisons were generated uniformly from 100 users and 100 items, and the "ground truth" embedding was generated on $[-1, 1]$ in d dimensions.

5.4.2 Distribution of errors in a generated embedding

In this experiment, shown in Figure 11, we demonstrate the distribution of the number of violations and ℓ_2 reconstruction error at each iteration is approximately normally distributed with a relatively small variance. This experiment demonstrates that the mean error shown in the previous experiments is shared relatively evenly between all points in the embedding, rather than being concentrated in a few outliers.

A relatively uniform error distribution is encouraging, suggesting that the limit to achieving better performance may be in the number of iterations through each point in the embedding or the number of useful comparisons present rather than a fundamental limitation of the algorithm. In contrast, if the mean error was caused primarily by a few points with large errors, we may have fallen into a local minimum that may be difficult to get out of.

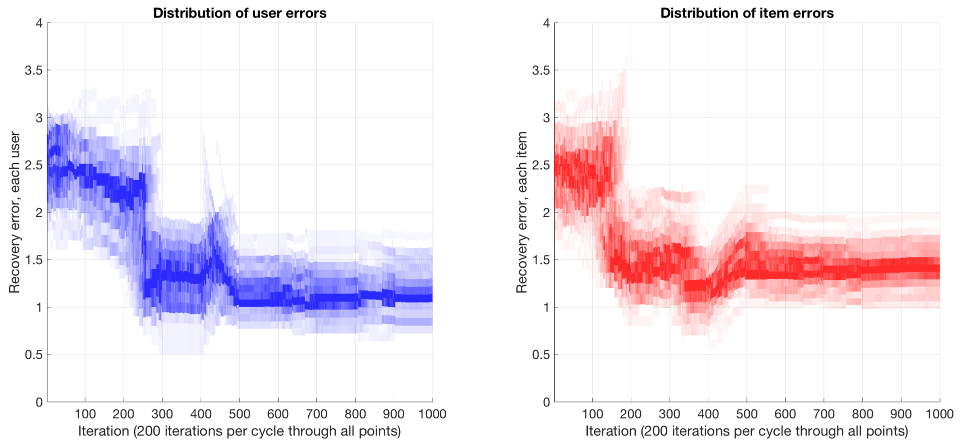


Figure 11: Distribution of errors for users and items in a generated embedding measured in terms of ℓ_2 recovery error, using the Procrustes distance to ignore the effects of a shift, scale, or rotation from the initial embedding. The comparisons were generated uniformly from 100 users and 100 items, and the “ground truth” embedding was generated on $[-1, 1]$ in $d = 20$ dimensions. Darker colors represent a higher number of errors in that bin.

CHAPTER VI

CONCLUSION

In this work, we considered an embedding of users and items that is useful in the context of a recommendation system following the ideal point model. We showed we can accurately localize new users and items using only a list of binary pairwise comparisons, and that even if the embedding contains a moderate level of noise, we can still localize a new user or item accurately.

Next, we extended our method for localizing a single point in the embedding to iteratively localizing every user and item and showed that we can use this to significantly reduce the average error of a noisy embedding. In the degenerate case, when we have no *a priori* knowledge of the embedding, we show that we can generate an accurate embedding using only the binary pairwise comparisons.

Experiments with synthetic data showed these results extend well to higher dimensionalities, suggesting that this method will apply well to recommendation systems in practice (such as predicting user preference in the MovieLens dataset) where embeddings must live in high dimensionalities to achieve acceptable performance.

There are several directions for future work:

- *Clever selection of initial embedding*: The experiments presented in Chapter V show significant accuracy increase in the second cycle through each point in the embedding, suggesting that a more clever choice of initial embedding may result in faster convergence time. Further, we show that when performing multiple tests with the same parameters but a different random seed results in differing performance. This suggests that the initial embedding plays a role in the accuracy we are able to achieve at "convergence."

- *Automatic selection of the* $\boldsymbol{c}$ *parameter*: The $\boldsymbol{c}$ parameter in the optimization problem of Equation 10, which controls the trade-off between the regularization and violations terms in the objective function, should be adapted to the confidence we have in each point in the embedding. One possibility for selecting this is by using sensitivity analysis, which is a measure of the sensitivity of the localized point to changes in the constraints. Geometrically, this corresponds calculating how the size of the feasible region responds to slight changes in the positions of other users and items.

- *Testing with data from application domains*: The MovieLens dataset is a popular metric for recommendation system applications. The MovieLens 100k dataset contains approximately 1000 users and 1400 items with 100k total ratings; the MovieLens 1M and 10M datasets contain many more ratings. Using these datasets will give us an avenue to compare this algorithm against other methods, such as 1-bit matrix completion.

- *Random perturbations and parallelization*: Although accuracy improvement after several cycles through the data is minimal, experiments have shown that adding small perturbations to every point in the embedding after each cycle may allow for accuracy improvements even after many cycles through the data (from an optimization perspective, our localization procedure is performing a block coordinate descent, and perturbations allow us to escape from local minima). In addition, each iteration may be performed in parallel, resulting in potential performance gains. Although each iteration currently depends on the updated value from every previous iteration, with a large number of iterations the effect of not considering the improvement from several previous iterations will be minimal. Using this simplification, the algorithm is "embarrassingly parallel," and parallel execution can result in a large speedup.

REFERENCES

[1] Boyd, S. and Vandenberghe, L., *Convex Optimization.* Cambridge University Press, 2004.

[2] Davenport, M. A., "Lost without a compass: Nonmetric triangulation and landmark multidimensional scaling," in *Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP), 2013 IEEE 5th International Workshop on*, pp. 13–16, Dec 2013.

[3] Grant, M. and Boyd, S., "Graph implementations for nonsmooth convex programs," in *Recent Advances in Learning and Control* (Blondel, V., Boyd, S., and Kimura, H., eds.), Lecture Notes in Control and Information Sciences, pp. 95–110, Springer-Verlag Limited, 2008. `http://stanford.edu/~boyd/graph_dcp.html`.

[4] Grant, M. and Boyd, S., "CVX: Matlab software for disciplined convex programming, version 2.1." `http://cvxr.com/cvx`, Mar. 2014.

[5] Maydeu-Olivares, A. and Böckenholt, U., "Modeling preference data," *The SAGE handbook of quantitative methods in psychology*, pp. 264–282, 2009.

Localizing Embeddings for Recommendation Systems using Binary Pairwise Comparisons

Matthew R. O'Shaughnessy

33 Pages

Directed by Professor Mark Davenport

Recommendation systems predict the preferences of users (who may be, for example, customers of an online shopping website or moviegoers), to various items (for example, consumer products or movies). One way this information is modeled is the ideal point model of preference, in which items and users live in an n-dimensional Euclidean space where each dimension represents an attribute and a small distance between an item and user indicates the user has a preference for that item. We seek to determine an embedding of many items and users given only binary pairwise comparisons of the form "user $\boldsymbol{x}$ prefers item $\boldsymbol{q_i}$ to item $\boldsymbol{q_j}$." First, we present an optimization-based framework for localizing new items and users given an existing embedding. We demonstrate that user localization can be formulated as a simple constrained quadratic program. Further, we show that although item localization produces a quadratically-constrained quadratic program which is difficult to solve, we can make the problem more computationally tractable by strategically combining comparisons to make the quadratic constraints into linear constraints. Finally, we show that by iteratively applying this localization method to every item and user, we can recover an embedding that agrees with almost every comparison, allowing us to iteratively improve the accuracy of a noisy embedding or even create an embedding using no a priori knowledge apart from the list of pairwise comparisons. Throughout, we present implementation details and optimization algorithms which make the recommendation system computationally efficient even with large datasets and dimensionalities.