Latent Variable Matrix Factorization in Heterogeneous Peer Prediction

Brian Hentschel Anna S. Hilgard Casey Meehan

Abstract

In the setting of heterogeneous peer prediction, soliciting truthful reporting from agent's is done via learning an agent's reporting type. This learned reporting type is used to produce payout schemes such that telling the truth is a low-regret strategy. Mostly, this work is done via clustering on user-user Δ Matrices, which capture dependence in the probability of user's reports. This clustering is done directly using matrix norms on the Δ Matrices.

In this paper, we present advantages to using latent variable matrix factorization in heterogeneous peer prediction. We show that clustering users by their corresponding latent variables is enough to produce nearly as good a clustering as using the Δ Matrices directly. Additionally, we show that it is orders of magnitude faster to produce these clusterings when using a latent variable representation. Finally, we show how latent variables and non-task data can be used in conjunction to produce solutions for the cold start problem, where a user has no report data and thus has an unknown initial reporting strategy.

1. Introduction

In peer prediction settings with heterogeneous users, the goal is to make truthful reporting a low-regret strategy for any agent. To fully minimize expected regret while allowing for heterogeneity, this would require the computation and use of individualized user-user signal correlations for every pair of users. This is in general computationally infeasible, so methodologies typically focus instead on clustering similar users while trying to minimally distort their individual signal distributions.

Additionally, because this method explicitly requires an existing signal distribution, there is a cold start problem: if we do not already have a significant amount of task data for a new user, we cannot know how properly to reward them in comparison to their peers. In our project, we focus on two specific goals:

1. Clustering is usually done by comparing each user's reports with other users. We aim to speed up this clustering by using latent variable matrix factorization and then clustering on much smaller latent variables.
2. Bootstrapping the clustering process by using non-task data to form an initial clustering before agent reporting begins.

2. Background and Related Work

In Shnayder et al. (2016), Shnayder et al. prove that the Correlated Agreement multi-task peer prediction mechanism maximally incentivizes truthful reporting without a ground truth for homogeneous users. The CA mechanism can be generalized to non-homogeneous users but requires a prohibitive number of user reports; thus, Agarwal et al. (2017) bounds expected regret by clustering users with similar signal-report distributions. This regret bound is proportional to the L1 difference between user-user 'Δ Matrices' and those of their respective clusters, where a Δ Matrix of users q, r is defined as:

$$D_{q,r}(i,j) = p(\text{q reports i, r reports j}) - p(\text{q reports i})p(\text{r reports j})$$

As far as we know, (Agarwal et al., 2017) is the only paper to specifically consider clustering methods for heterogeneous peer prediction. The paper differs from our work in that it assumes all of the relevant data exists and seeks to prove a theoretical upper bound on regret rather than to study empirical average regret in settings with sparse data. To demonstrate this, the authors restrict their dataset to a small subset of only those users for whom there is a full joint distribution between any two users, resulting in only 269 users.

Further, while the authors prove a theoretical bound in the case of empirically estimated ground truth, the tests they run on real data rely on the existence of a known ground truth label. This allows them to calculate individual signal confusion matrices using methods developed in Dawid & Skene (1979) and leverage tensor decomposition techniques from Anandkumar et al. (2014) and Zhang et al. (2016) to compute average clusters directly. We seek to show that in the more general case of incomplete data with no ground truth labels, clustering instead on user-specific

latent variables learned through matrix factorization can lead to similar computational savings as tensor decomposition while eliminating the need for labels. Further, we show that these computationally efficient clusterings are similarly effective with respect to incentivizing truthfulness as clustering on Δ matrices directly.

The use of the Correlated Agreement mechanism (Shnayder et al., 2016) requires that the Δ matrices are already known or can be empirically estimated from user reports. However, in practice, newly introduced users have contributed no reports and thereby cannot be clustered accurately. While nothing in the peer prediction literature addresses this cold start problem, it is well studied in the field of matrix factorization. The most similar method to ours, in that it incorporates non-task data, is (Kula, 2015). In this model, the author attempts to combine collaborative filtering and content-based recommendation systems by encoding item and user latent vectors as sums of vectors for the non-task features to which they correspond. Non-task features of an item or user in this model correspond to binary indicators of whether or not text labels describe the item or user. For example, in the setting of retail recommendations, features of a given item might be 'pencil skirt' and 'blue'. Features of a user are text character descriptions drawn from user profiles. In our expansion on this model, we allow non-binary user features from the results of psychology and aptitude tests. We also allow for a portion of the user latent vectors to be unexplained by non-task user features and learned over time as tasks are completed.

3. Datasets

3.1. Task Data

The primary dataset we use is that of the *Good Judgment Global Forecasting Competition*, available at `https://dataverse.harvard.edu/dataverse.xhtml?alias=gjp`. There are many challenges inherent to this dataset. First, the idea of a task is not well-defined. Users enter predictions in continuous time, and the true probability of an event will change over time, such that each minute could be considered a different task (Note that if we expect users to only update their predictions in response to new information, these time-question tasks will be independent conditional on the signal). We choose to bucket these responses by week to mediate between the desire to allow for variation in true probability over time while maintaining sufficient task response density. This results in 1,499,294 predictions by 2,052 users on 1,548 tasks. Secondly, the signal itself can be expected to have a high degree of noise, as the probability of any of these global events is not well-defined even in the absence of individual interpretation effects.

To study the effects of the clustering methodologies in a slightly less noisy realm, we additionally use a second dataset of user judgments on whether or not websites contained adult content. This dataset, termed *Adult*, was pulled from the Square Project at the University of Texas. The data is available at `https://github.com/ipeirotis/Get-Another-Label/tree/master/data`. We limit the dataset to users who gave at least one report of each signal, resulting in 65,144 reports by 389 users on 10,381 tasks.

Both datasets required significant preprocessing to account for sparsity of signal distributions and to compute the Δ matrices. Additionally, because Good Judgment reports were continuous rather than discrete, we bucketed the responses into five uniform bins, which we use to compute the matrices.

3.2. Non-Task Data

In addition to user predictions (task data), the *Good Judgment* dataset includes a variety of questions to assess the knowledge and ability of each user as well as their potential psychological biases. For instance, questions to assess knowledge might ask them about world events or to solve a math problem. Questions about character might ask them to state on a scale from 1-7 how just they believe the world is. We then use this non-task data to address the cold start problem discussed in section 2.

The psychological data contains both binary and ordinal features. A binary feature may be an answer to a 'yes/no' question or whether the user correctly answered a diagnostic math test question. An ordinal feature captures some continuous measure of that user's character, *e.g.* user age or weekly workhours. To discretize ordinal features, we bucket each into a finite set of uniform bins similar to the Good Judgment report data. However, the number of bins is feature-dependent.

The dataset consists of 168 features for each of the 2,052 users. The final set of 168 features consists of only features for which most users have data; the data set contains approximately another 1000 features, most of which are very sparsely populated. Even using the most densely populated features, there are a significant number of missing answers in the dataset. For binary features, we made negative answers into -1, missing answers into 0, and positive answers into 1. For ordinal features, we filled the missing answers in with the average value of that feature.

4. Model

Given a sparse matrix containing tasks, users, and users' reports on each task, the goal is to produce clusters which ac-

curately describe users' reporting behavior. Because of the sparse matrix data, we focus on matrix factorization models to generate latent factor representations of users. That is, we assume that there is some latent variable representation of a user's forecasting behavior, u_i that interacts with related task-specific factors, t_j. These could be thought of, for example as the task's topic areas and the user's expertise or bias in those specific topic areas.

4.1. Good Judgment Matrix Factorization Model

After transforming the predictions to range from -1 to 1 instead of 0 to 1, we use the following model, shown in Figure 1:

$$p_{i,j} \sim \mathcal{N}(\tanh\left(\mu_{i_M}\left[u_i^T t_j + g + \mu_j\right] + \mu_{i_A}\right), \sigma^2)$$

Users tend to exhibit two forms of individualized biases in this space. The multiplicative bias, μ_{i_A} captures either risk aversion or overconfidence, such that they tend to predict values either closer to or farther from 0 (.5) than are warranted by their knowledge. The additive bias, μ_{i_M} captures both availability bias or normalcy bias which would result in user having a tendency to predict a specific level. The task bias μ_j captures the general prediction likelihood of the task absent user-specific factors, and the global bias g captures the overall mean of the user-task predictions. The tanh restricts the space of the predicted mean to the domain of the target value.

Then, we must infer $k + 2$ parameters for each user, $k + 1$ parameters for each task, and one global bias, where k is the length of the latent vector representation we choose for users and tasks. Only the predictions $p_{i,j}$ are known in this model.

For *Adult*, which does not have this probablistic representation and has reports ranging from 1 to 4, we use only additive biases and remove the tanh.

4.2. Using Non-Task Data to Extend the Model for Cold Start Users

To encourage truthful reporting in a user who has not yet completed a significant amount of task data, we would like to be able to predict an appropriate clustering using non-task data. The Δ matrix representation of a user has over 50,000 features for a user, given that a separate Δ matrix must be estimated for the signal-report distribution of a user with every other user he/she could be scored against. This is infeasible to predict from a limited amount of pre-task information. However, the *Good Judgment* matrix factorization model encodes users in as few as 4 parameters in our model. This allows us to attempt to use non-task psychological data to predict a user's latent variables. We choose this approach over predicting a user's cluster directly as it leads to an intuitive distance metric for training and provides a more interpretable representation of how confident the model is in a given prediction.

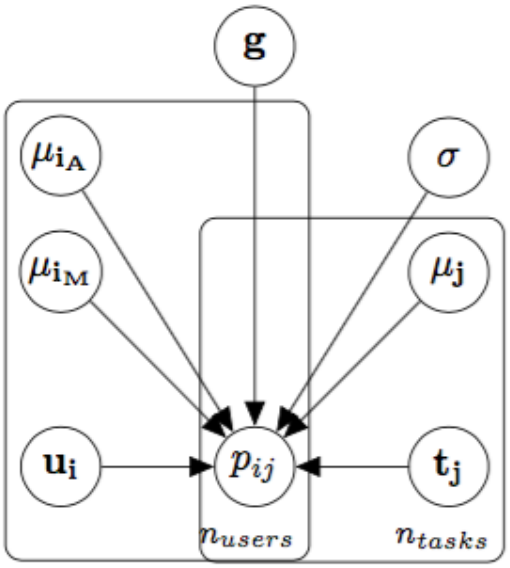

Figure 1. Model of prediction generation.

We focus only on the prediction of user latent vectors from non-task data. Similar techniques could be used to solve the cold-start problem for tasks, however this is not relevant to our specific use of the system. For predicting the user latent variable models from psychological data, we try two separate techniques:

1. Directly predict latent variables from psychological data with no task data:
In our first model, we try predicting a user's latent variable representation directly from his or her psychological data. We use both a more interpretable simple linear regression model and a less interpretable fully connected neural network to perform the predictions. When using the predicted latent variables online, it would be possible to adapt to incoming task data over time by using a weighted combination of the predicted latent variables and the trained latent variables from matrix factorization with the former decaying as the quantity of task data increases.

2. Create user latent variable representation by summing feature latent variables In this model, adapted from (Kula, 2015), a user's latent variable representation is created through a weighted summation of psychological feature latent variables. Each psychological feature, like 'age' and 'math_score', has a latent vector representation. In a cold-start, the user's latent representation is the sum of those psychological feature vectors, each weighted by the user's responses to that psychological question. So, the

'age' feature vector is weighted by an individual's (binned) age value before being summed into their latent representation. As their reports become available, an additional user-specific latent vector is added, such that in absence of feature latent vectors the model simplifies to matrix factorization.

More formally, let f be a set of features, each with a latent vector $\mathbf{L_f}$. For user i with feature values u_{if}, the user's latent variable representation is

$$u_i = \sum_{i=1}^{|f|} u_{if}\mathbf{L_f} + \mathbf{L_{u_i}}$$

where $\mathbf{L_{u_i}}$ is a user-specific latent vector which starts at $\mathbf{0}$. So, a user's cold-start latent representation is simply $\sum_{i=1}^{|f|} u_f\mathbf{L_f}$, as the user-specific latent variable is $\mathbf{0}$ without reports. As reports from the user become available, the user-specific latent vector $\mathbf{L_u}$ moves away from $\mathbf{0}$. Predictions of the user's report are then made as in the Matrix Factorization model through

$$p_{i,j} = \tanh\left(\mu_{i_M}\left[u_i^T t_j + g + \mu_j\right] + \mu_{i_A}\right)$$

Because user's have a non-zero latent vector prior to producing any reports, they can be given an initial clustering. We test how good this initial clustering is by comparing the distance between these cold-start latent representations to the data-complete latent representations (with user reports).

5. Training

5.1. Matrix Factorization for Clustering

For learning the latent user parameters, we use mini-batch stochastic gradient descent to find the maximum likelihood estimate. We choose not to use a prior for regularization as we find that the training and validation performance do not generally diverge. This is likely because there is already enough noise in the training data for each parameter that it is difficult to overfit the model. We do not explicitly code the initialization of the model, so of course it is possible it finds a local rather than global minimum. However, the performance of the model is consistent across random starts.

5.2. Predicting User Representations: Hybrid CF/CB Approach

In the above Algorithm 1, parameters include user and task specific biases, the global bias, the latent vectors for tasks, and the latent vectors for specific users as well as user features. In (Kula, 2015), the user feature latent vectors were trained at the same time as user specific latent vectors. However, for reasons described below, we find that

Algorithm 1 Mini-Batch SGD Parameter Fitting

```
learning rate = α
for epoch in epochs do
  for batch in batches do
    Calculate the gradient of the MSE Loss for each parameter for all users and tasks in the batch
    Update the parameters:
    param ← param - α * d/dparam(MSE Loss)
  end for
end for
```

it makes more sense to train the latent variable model first only for user feature latent vectors. Then, after locking the user feature latent vectors and a corresponding set of task parameters: t, we then train a set of user specific latent vectors. Reports for new users are predicted as

$$p_{i,j} = \tanh\left(u_i^T t_j + g + \mu_j\right)$$

where u_i is formed as described in Section 4.2. Reports for all other users are predicted as

$$p_{i,j} = \tanh\left(\mu_{i_M}\left[u_i^T t_j + g + \mu_j\right] + \mu_{i_A}\right)$$

The global bias g and task-specific bias μ_j were allowed to vary throughout both training stages, although their values did not change much from stage 1 to 2.

There are two reasons that we train the user feature vectors before the individual user biases. First, because the model incorporating user feature biases is more general than the original matrix factorization model, it has more parameters and can be sensitive to the learning rate. In particular, we found that training all at once provided a worse RMSE on the predicted report values than using the original matrix factorization model, with the former having a RMSE of 0.33 and the latter having an RMSE of 0.28. This is most likely due to a too high learning rate, but the model already takes a significant time to converge at the high learning rate. If we train the user feature vectors first and then train the individual user latent vectors, the final RMSE was 0.24.

Additionally, the model above does not achieve a unique global minimum. To see this, assume we are at some global minimum and there exists binary feature f. Then consider subtracting some k dimensional vector a_k from the feature latent vector $\mathbf{L_f}$ and then adding the same vector to each user latent vector L_{u_i} with u_i having feature f. Then, $u_i = \sum_{i=1}^{|f|} u_{if}\mathbf{L_f} + \mathbf{L_{u_i}}$ is the same for each user, and so we still have the same globally minimum error. Thus, we seek to train our data in such a way that the user feature latent vectors are informative as possible. One way to achieve this is to train only the user feature latent variables first and minimize loss, and then to train the individual user latent vectors after.

5.3. Predicting User Representations: Linear Regression and Neural Network Approach

Both the linear regression and the neural network are trained using out of the box PyTorch libraries. For each, we use stochastic gradient descent as our optimizer with a mean squared error loss on the predicted latent variables.

6. Methods

For the *Good Judgment* Dataset, we use the file `survey_fcasts.yr4.tab`, consisting of all user forecasts for the fourth year of the tournament. The file has 1,685,784 lines, each consisting of details about a specific user id and forecast. We restrict to only binary valued questions, for which a single probability of 'yes' is given as the answer (as opposed to, for example, "Who will be inaugurated as President of Russia in 2012?", which had 3 possible answers). We restrict to only users who responded to at least 30 Individual Forecasting Problems (IFPs) over the course of the year. Given that users can (and ideally do) update their predictions for IFPs over time, any given time period could be thought of as a new IFP. Following the scoring procedure of the Good Judgment, we assume that if a user enters a prediction and then does not update it, this implies that they would like their existing prediction to be carried forward until they do update. We allow each week an IFP is active to be a different prediction task to balance temporal variability in task signals with our desire for higher task report density.

In computing the user latent vectors, we transform each prediction p from the range $[0, 1]$ to $[-1, 1]$ via the transformation

$$p = (p - 0.5) * 2$$

. This puts the predictions over the entire range of the tanh function and makes it so the the multiplicative user bias smoothly transforms data towards predictions of 1 or -1 (as opposed to outside the range). We train with stochastic gradient descent for 75 epochs with a learning rate of .2.

6.1. Clustering

As a metric of the clustering success, we use the average element-wise L1 distance between clustered Δ matrices and the original user-user Δ matrices for each pair of users. This requires binning the continuous predictions into discrete buckets. We choose to use 5 uniformly spaced bins from 0 to 1. We then compute the empirical joint distribution for any two users based on these discrete signals, as well as the empirical signal distributions of each individual. As mentioned in Section 2, the Δ matrices can be calculated from these as:

$$\begin{aligned} D_{q,r}(i,j) = p(&\text{q reports i, r reports j}) \\ &- p(\text{q reports i})p(\text{r reports j}) \end{aligned}$$

To cluster, we use the scikit-learn implementation of K-means on all cores. The baseline we compare to is that of clustering on the Δ matrices directly, which is the method described in Agarwal et al. (2017) and should in general be better at guaranteeing similarity of Δ matrices than clustering on other factors. In regards to user coordinates for clustering, we append all of the Δ matrices between that user and all other users, leaving NaNs where not enough data exists to compute a distribution, and flatten this into a vector. When clustering, we iteratively impute coordinate means for NaNs such that they are not taken into consideration in the clustering.

The *Adult* Dataset has 92,720 lines, each of which has a report by a user id on a website. Ratings are discrete and ordinal, and we use them as given for both learning latent user variables and for computing Δ matrices.

6.2. Cold Start Problem

As described in Section 5, in the cold start problem we focus on predicting latent variables instead of predicting clusters directly. In technique 1, we predict latent variables from user psychological testing data. These supervised methods takes as labels the results of the original matrix factorization model for the given users, so that we have labeled data for each user of f(user id, psychological data) $\rightarrow$ (latent vector representation). To estimate the function f, in the first technique we performed simple linear regression over the psychological features onto the latent vector representations. In the second case, we trained a fully connected neural network. We tried 3 and 4 layers for the network, and varied the number of internal nodes between 20 and 80 for both layers. We found the best performance with a shallow network of just 3 layers and a middle layer of 20 nodes. Both techniques use out of the box PyTorch libraries.

For the hybrid CF/CB technique of using user feature dependent latent vectors, we implemented the model using PyTorch. The model was then trained using stochastic gradient descent, with a learning rate of 0.2. We took 75 iterations to train the matrix factorization unless otherwise noted.

7. Results

7.1. Sophie

While we achieved the lowest training and validation loss using 10 latent user factors (See Figure 2), the best L1 loss against the original Δ matrices was achieved by clustering

Table 1. Distribution of L1 Distances Across Clusterings

	Δ (100)	K=2 (100)	Δ (500)	K=2 (500)
mean	.0168	.0215	.0145	.0192
std	.0243	.0290	.0218	.0258
min	0	0	0	0
25%	.0021	.0043	.0015	.0036
50%	.0084	.0108	.0071	.0099
75%	.0216	.0271	.0186	.0244
max	.7118	.6864	.6536	.6191

Table 2. Runtime (s) of KMeans algorithm on User Representation

Clusters	Δ	K=2	K=3	K=4
100	423	.57	.55	.65
150	625	.66	.66	.66
200	800	.66	.77	.87
250	959	.87	.88	.97
300	1187	.96	1.1	1.1
350	1198	1.1	1.2	1.2

on only 2 latent user factors (See Figure 3). Distribution results are given in Table 1.

However, clustering on the latent factors for all values of K was significantly faster than clustering directly on the Δ matrices, as the flattened concatenated matrices result in 51,300 features for each user. See Table 2 for details.

7.2. Cold Start Problem: Predicting Latent Variables Directly

Table 3 shows the performance using a Neural Network and simple linear regression in predicting the latent variables for new users from just psychological data. The mean distance is calculated as the ℓ_2 distance from their final trained values after observing predictions and updating their latent vectors. Both are compared to just using a latent vector made up of coordinate-wise mean values for each new user. That is, the mean vector takes each of the k dimensions for a latent vector and estimates a new user's value in that dimenion by the average of every other user's value for that dimension.

For both linear regression and the neural network, there is some improvement over just using the mean latent vector, and so we see some improvement over a random guess for new users. Ideally, this effect would be much larger; the extent to which this is possible using the psychological data for Good Judgment is unknown. Indeed, considering that there was no prior method of predicting new user's reports for Good Judgment, this provides some utitlity.

Table 4 shows the results for a similar experiment using the feature-based expansion of matrix factorization. In this

Table 3. Cold Start Predictions

Method	Mean Distance
Baseline (Mean Latent Vector)	0.67
Linear Regression	0.58
Neural Network	0.56

Table 4. Cold Start Predictions

Method	Mean Distance
Baseline (Mean Latent Vector)	0.61
From Sum of User Feature Vectors	0.53

second set of results, we look at a user's initial latent vector from its features and calculate the ℓ_2 distance from its final latent variable u_i after training. Similarly to before, we compare this to using the mean latent vector for a user. We note that this is not in the same table as Table 3 because the corresponding set of task latent vectors is different. Similarly to before, we are given a small headstart on predicting a user's final latent vector from just these psychological values. In addition, this technique is easier to integrate into a full system as its easy to see how to make online updates to a user's latent vector representation. In the previous technique, it's unclear how to best transform from the initial prediction of a user's latent vector to its trained representation. In this technique, this happens naturally as a result of training.

8. Discussion

The results show us that it is possible to use a significantly compressed representation of user heterogeneity and still arrive at a clustering that is still almost as successful at encouraging truthful reporting as the significantly more time- and memory-intensive task of clustering directly on the user-user signal distributions. Notably, the user representation we use does not explicitly encode the interactions between any two users‘ reporting behaviors yet performs nearly as well as the clustering will full knowledge of this data.

We note that the RMSE and L1 loss of the training model are lowest for larger numbers of latent factors, yet the clustering performs best when only two latent factors are used. One possible explanation is that this setup forces more of the user variability into the multiplicative and additive biases, which are more specifically relevant to the task of predicting a distribution of reports given signals. Supporting this interpretation, clustering the *Adult* dataset, which does not have the task structure to allow for modeling known psychological biases (specifically with respect to probability reporting) as its latent user biases, does not degrade as we increase the number of latent factors.

TODO INCLUDE CHART

9. Conclusion

- What happened?
- What next?

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

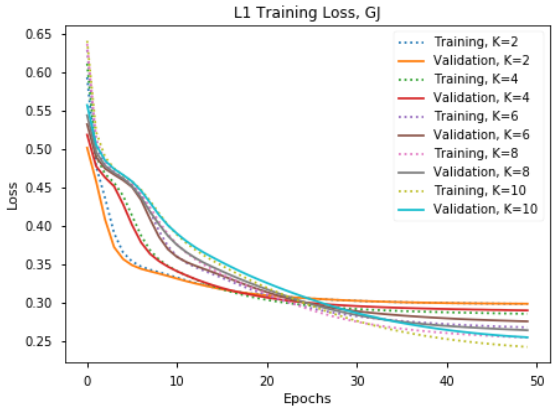

Figure 2. Training Results, L1 Loss.

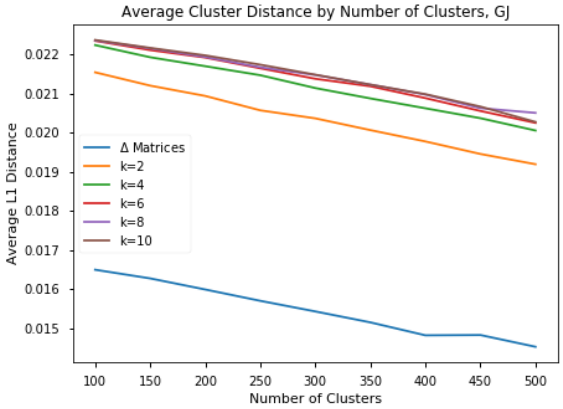

Figure 3. Average L1 Distance.

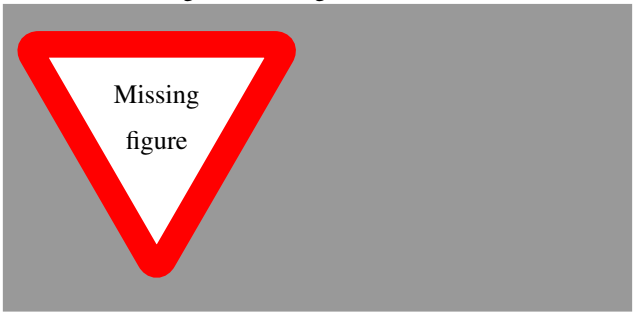

Figure 4. Visualizations of the internals of the system.

References

Agarwal, Arpit, Mandal, Debmalya, Parkes, David C, and Shah, Nisarg. Peer prediction with heterogeneous users. 2017.

Anandkumar, Animashree, Ge, Rong, Hsu, Daniel, Kakade, Sham M, and Telgarsky, Matus. Tensor decompositions for learning latent variable models. *Journal of Machine Learning Research*, 15:2773–2832, 2014.

Dawid, Alexander Philip and Skene, Allan M. Maximum likelihood estimation of observer error-rates using the em algorithm. *Applied statistics*, pp. 20–28, 1979.

Kula, Maciej. Metadata embeddings for user and item cold-start recommendations. *arXiv preprint arXiv:1507.08439*, 2015.

Shnayder, Victor, Agarwal, Arpit, Frongillo, Rafael, and Parkes, David C. Informed truthfulness in multi-task peer prediction. In *Proceedings of the 2016 ACM Conference on Economics and Computation*, pp. 179–196. ACM, 2016.

Zhang, Yuchen, Chen, Xi, Zhou, Dengyong, and Jordan, Michael I. Spectral methods meet em: A provably optimal algorithm for crowdsourcing. *The Journal of Machine Learning Research*, 17(1):3537–3580, 2016.