Co-Clustering by Bipartite Spectral Graph Partitioning for Out-of-Tutor Prediction

Shubhendu Trivedi
Dept. of Computer Science
Worcester Polytechnic Institute
Worcester, United States
shubhendu_trivedi@ieee.org

Zachary A. Pardos
Dept. of Computer Science
Worcester Polytechnic Institute
Worcester, United States
zpardos@wpi.edu

Gábor N. Sárközy
Dept. of Computer Science
Worcester Polytechnic Institute
Worcester, United States
gsarkozy@cs.wpi.edu

Neil T. Heffernan
Dept. of Computer Science
Worcester Polytechnic Institute
Worcester, United States
nth@cs.wpi.edu

ABSTRACT

Learning a more distributed representation of the input feature space is a powerful method to boost the performance of a given predictor. Often this is accomplished by partitioning the data into homogeneous groups by clustering so that separate models could be trained on each cluster. Intuitively each such predictor is a better representative of the members of the given cluster than a predictor trained on the entire data-set. Previous work has used this basic premise to construct a simple yet strong bagging strategy. However, such models have one significant drawback: Instances (such as students) are clustered while features (tutor usage features/items) are left alone. One-way clustering by using some objective function measures the degree of homogeneity between data instances. Often it is noticed that features also influence final prediction in homogeneous groups. This indicates a duality in the relationship between clusters of instances and clusters of features. Co-Clustering simultaneously measures the degree of homogeneity in both data instances and features, thus also achieving clustering and dimensionality reduction simultaneously. Students and features could be modelled as a bipartite graph and a simultaneous clustering could be posed as a bipartite graph partitioning problem. In this paper we integrate an effective bagging strategy with Co-Clustering and present results for prediction of out-of-tutor performance of students. We report that such a strategy is very useful and intuitive, even improving upon performance achieved by previous work.

Keywords

Out-of-Tutor Prediction, Dynamic Assessment, Spectral Co-clustering, Ensemble Learning, Bootstrap-Aggregation

1. INTRODUCTION

A significantly large student population would usually have a wide variation in learning rates and knowledge levels. While there are numerous reasons for this diversity, three major reasons are related to: the type of instruction or help they respond best to, the way they are oriented towards learning and their levels of intellectual development [1],[2]. Needless to say, such differences would be reflected in the way students interact with educational software, making educational data quite difficult to mine well. Specifically there are many educational data mining problems where the end goal is to predict the performance of a student on a given in-tutor or out-of-tutor task. In-tutor tasks include predicting the probability that a student will answer an item correctly after attempting a sequence of similar questions whereas out-of-tutor tasks include being to predict student performance in post-tests based on the data from their tutor usage.

The idea that students are quite different makes it apparent that perhaps it is not such a good idea to fit a global prediction model over the entire dataset for making predictions. In spite of the differences between students, educators commonly observe that students actually lie in very rough groups and have similar pedagogical needs. Taking a cue from this intuition, the task of prediction can be improved by clustering students into somewhat homogeneous groups and then training a separate predictor for each group. Such a predictor would obviously be a much better representative of students in that cluster as compared to a predictor which is fit on the entire dataset. For example, it makes sense to have a different model for students roughly classified as fast learners and a different model for slow learners than the same for both. This rather simple strategy of grouping students together and then modeling them separately can lead to improved performance in prediction and perhaps even better interpret-ability.

While the above approach is compelling, there are two major issues with it. Firstly, while it is useful to model students as belonging to different groups, it is also known that such groupings are quite fuzzy and approximate. Students might actually possess different characteristics in varying degrees and what really sets them apart are certain dominant characteristics. For example students classified as fast learners might actually be slow learners in certain skills. A fast learner might also belong to the group of students that are good at recalling information etc. Thus, such complex characteristics can not be possibly modelled by simply clustering

students to a certain limit and then training models for each cluster. This "spread" of features in a student across groups also needs to be captured to make a distributed predictive model such as the above more meaningful. Such an issue can be resolved by varying the granularity of the clustering and training separate models each time so that such features can be accounted for. A simple yet quite effective strategy to do so was proposed by the authors and was seen to work quite well both in educational contexts (in-tutor predictions [3], out-of-tutor predictions [4],[5]) and more generally [6].

The second problem with the above approach is that clustering is implicitly suggested to be one-way i.e only clustering students. But this need not necessarily be the case and only clustering students would consider only half of the story. As an example, consider a matrix in which the rows represent students and the columns represent their responses to certain items. Clearly, clustering students would depend upon their item distributions, implicitly suggesting that for certain students certain items are more important than others. Similarly if items were to be clustered, they would depend on which groups of students get them correct (or incorrect) most frequently. This indicates a duality between these two clusterings, which on simultaneous co-clustering could be very useful in answering many research questions. Co clustering of such a student versus item matrix would pair clusters of student proficiency to clusters of item performance which could be seen as a sort of a subject treatment interaction. This idea could be extended to the more general case of students and features rather than just items. In this work we use this idea of co-clustering students and their tutor interaction features and interleave it with the bagging strategy which was used with clustering [3],[4],[5],[6]. This combined approach is then used to predict the post-test scores of students.

This paper is organized as follows: In Section 2 we discuss the idea of co clustering in more detail and that co clustering could be posed as a bipartite graph partitioning problem. In Section 3 we describe a general framework in which we interleave co clustering with the idea of generating an ensemble. In Section 4 we describe the experimental results which demonstrate the validity of this approach. In Section 5 we discuss the results and also describe some avenues for further work.

2. CO-CLUSTERING

Clustering is a fundamental tool from unsupervised learning for data analysis that groups together relatively homogeneous objects. The central idea for clustering is that every object could be specified by a feature vector (or a point in the feature space) and then the degree of homogeneity between them could be measured by some objective function that uses these feature vectors. For example in k-means clustering: the points are grouped so as to minimize a distortion function, which is basically the sum of distances of all points from their assigned cluster centroids [7].

Clustering algorithms are one-way, i.e. one dimension of the data (say the rows of the data matrix) is clustered based on the similarities measured on the second dimension (say the columns). As pointed out in the previous section it might be desirable, quite frequently, to cluster along both the dimensions simultaneously, exploiting the apparent duality between them. Such simultaneous clustering can often offer interesting insights about the nature of interaction between the clusters at both the dimensions [8]. This utility is fast making co-clustering a fundamental tool for data analysis as is indicated by its widespread use in text and document mining [9], [10]; bioinformatics and gene expression analysis [11], [12]; collaborative filtering [13] and many others practical applications.

While there are now a number of approaches to co-clustering such as based on spectral graph theory [10] and information theory [14], [15], each with its advantages, we consider the approach proposed by Dhillon [10] which formulates the problem of co-clustering as a bipartite graph partitioning problem. We now briefly describe this approach starting with the relevant notation and definitions.

2.1 Notation and Definitions

A graph is represented as $G = (\mathcal{V}, E)$ where $\mathcal{V}$ represents the set of vertices and E represents the set of all edge weights E_{ij}, where E_{ij} is the edge weight between vertices $\{i, j\}$.

Definition 1. The $n \times n$ **Weighted Adjacency Matrix** of an undirected graph is defined as the matrix $(m_{ij})_{i,j=1,\ldots,n}$. If $m_{ij} = 0$ it implies that vertices v_i and v_j are not connected by an edge. If $m_{i,j} \neq 0$ it implies that the vertices $\{i, j\}$ are connected and $m_{i,j}$ is the corresponding edge weight. Since the graph is undirected, $m_{ij} = m_{ji}$ necessarily.

Definition 2. Given the weighted adjacency matrix of a graph and a partition of the vertex set $\mathcal{V}$ into two disjoint subsets $\mathcal{V}_1$ and $\mathcal{V}_2$, the **cut** between these two subsets is defined as:

$$cut(\mathcal{V}_1, \mathcal{V}_2) = \sum_{i \in \mathcal{V}_1, j \in \mathcal{V}_2} M_{ij}$$

An undirected **bipartite graph** is a triple represented by $G = (\mathcal{S}, \mathcal{F}, E)$ where $\mathcal{S}$ and $\mathcal{F}$ are two sets of vertices and E is the set of edges. Since it is a bipartite graph one end of the edges in set E have an endpoint in $\mathcal{S}$ and another in $\mathcal{F}$. In our case the set $\mathcal{S}$ is the set of students while the set $\mathcal{F}$ is the set of features. The set of features could readily be seen as a set of item-responses as well. If $\mathcal{F}$ is the set of items, then an edge between s_i and f_j exists if that item was answered correctly by a student and not otherwise. More generally, if $\mathcal{F}$ is just a set of features, then the edge $\{s_i, f_i\}$ simply represents the value of that feature scaled between 0 and 1 for that student. Given this definition of a Bipartite Graph, now we define the adjacency matrix of the same.

Consider a $m \times n$ dimensional data matrix with students on the rows and the items or features on the columns. Let's suppose this matrix is given by A. Clearly, the adjacency of the bipartite graph is given as:

$$M = \begin{bmatrix} 0 & A \\ A^T & 0 \end{bmatrix}$$

The zeroes on the top-left and the bottom-right sub-matrices signify the absence of connections amongst the elements of $\mathcal{S}$ and $\mathcal{F}$ respectively (since connections in a bipartite graph can only run between $\mathcal{S}$ and $\mathcal{F}$). The matrix M is represented such that taking A at the top right corner and A^T at the bottom left implies that the first m rows of M represent the set of students and the next n rows represent the set of features or items.

Suppose the Bipartite Graphs (whose adjacency matrix is defined above) is partitioned into k clusters $\mathcal{V}_1, \ldots, V_k$. Given this partitioning, a corresponding set of student clusters $\mathcal{S}_1 \ldots \mathcal{S}_k$ and corresponding feature clusters $\mathcal{F}_1 \ldots \mathcal{F}_k$ would also be obtained. It could be intuitively seen that the best possible such set of clustering for all such pairs would be when the sum of all edges which cross between clusters is the minimum possible. As defined by [10] this corresponds to:

$$cut(\mathcal{S}_1 \cup \mathcal{F}_1, \ldots, \mathcal{S}_k \cup \mathcal{F}_k) = min_{\mathcal{V}_1,\ldots,\mathcal{V}_k} cut(\mathcal{V}_1, \ldots, V_k)$$

Where $\mathcal{V}_1, \ldots, \mathcal{V}_k$ represents a k-partitioning of the graph. The above definition leads us to the Bipartite Graph Partitioning problem:

Definition 3. **The bipartite graph partitioning problem:** Given a graph as defined earlier and subsets of $\mathcal{V}$ which are almost of equal size, say $\mathcal{V}_1^*$ and $\mathcal{V}_2^*$. The required partition is

$$cut(\mathcal{V}_1^*, \mathcal{V}_2^*) = min_{\mathcal{V}_1,\mathcal{V}_2} cut(\mathcal{V}_1, \mathcal{V}_2)$$

The bipartite graph partitioning problem as defined above is NP-Complete. However, a good relaxation to this problem is given by spectral graph bi-partitioning. This relaxation is achieved via the graph Laplacian. The laplacian L of a graph is a symmetric positive semi-definite matrix such that its un-normalized form is given by $L = D - M$ where D is the degree matrix and M is the adjacency matrix as defined earlier. Note that D is only a diagonal matrix while M is a symmetric matrix with all zeros in the diagonal. Thus, the Laplacian encodes both D and M in it and has many useful properties such as being positive semi-definite, which make it very useful for tasks such as clustering [24]. One property of the Graph Laplacian that make it particularly suitable for clustering are related to the properties of its spectrum. The spectra of the Graph Laplacian unfolds the data manifold to give an lower dimensional embedding which can give "better" clustering results.
Returning to the Bipartite Graph Partitioning Problem, as demonstrated by Dhillon [10] and Mohar [24], the second eigenvector of the generalized eigenvalue problem $Lz = \lambda Dz$ gives a real relaxation to the problem of finding the minimum normalized cut $\mathcal{Q}(\mathcal{V}_1, \mathcal{V}_2)$. The normalized cut is basically a cut that favours finding balanced partitions i.e. if the cut of two different partitions is the same, then the normalized cut is smaller for that partition which is more balanced. Thus it favours partitions that are balanced and have a small cut value. Clearly, the normalized cut is more suitable for tasks such as clustering [16]. Note that this relates to the ideas above relating to the optimal bi-partitionings in the following way: We want balanced clusterings with minimum cut for solving the bipartite graph partitioning problem, which would also be the optimal clustering for us. Thus looking at the Laplacian of the bipartite graph might provide such a clustering.

2.2 Spectral Co-Clustering

Given the definitions and notions in the previous section, in this section we state an algorithm [10] for finding the optimal co-clusters $\{\mathcal{S}_1 \cup \mathcal{F}_1\}, \ldots, \{\mathcal{S}_k \cup \mathcal{F}_k\}$ as mentioned above. For that we define the graph laplacian of a bipartite graph as such an optimal clustering can be found using a laplacian. Using the definition of $L = D - M$ as defined above and also the definitions of D and M. The laplacian may be written as:

$$L = \begin{bmatrix} D_1 & -A \\ -A^T & D_2 \end{bmatrix}$$

and

$$D = \begin{bmatrix} D_1 & 0 \\ 0 & D_2 \end{bmatrix}$$

where D_1 and D_2 correspond to the degree matrices of A and A^T respectively.
If the generalized eigenvalue problem $Lz = \lambda Dz$ is written for the above laplacian for a bipartite graph and then rearranged, it has been demonstrated [10] that the resulting equations define the equations for a singular value decomposition of the normalized matrix

$$A_n = D_1^{-1/2} A D_2^{-1/2}$$

Thus instead of finding the second smallest eigenvector corresponding to the second eigenvalue, one could find the left and the right singular values in its place. Finding the right singular value gives a bi-partitioning of students while the left singular value gives a bi-partitioning of the features. These can then be used to find the optimal bi-partition as defined above.

Algorithm 1.

1. *Given the co-occurrence or data matrix scaled to between* 0 *and* 1 *A, form the normalized matrix.*

$$A_n = D_1^{-1/2} A D_2^{-1/2}$$

2. *Compute the second left and right singular vectors for A_n, concatenate them together to form a vector z.*

3. *Run k-means on this vector to obtain a simultaneous clustering of both the students and the features.*

This algorithm can be extended to a multipartition case if instead of finding the second singular values, the first $log_2(k)$ singular vectors are found. The rest of the process remains the same.
Note that this algorithm gives a simultaneous clustering of the rows and the columns and is restricted in the sense that the number of row and columns clusters have to be the same. We modify this by running k-means two times. If the number of row clusters is k and then the number of column vectors is l, then we run k-means on the vector z twice, once to find k clusters and then to find l clusters. The first m elements of the length $m + n$ cluster assignment vector run will then correspond to the row clusters and the last n elements of the cluster assignment vector in the second run will correspond to the column cluster indices.

3. BAGGING STRATEGY

The statement of the supervised learning problem in machine learning could be roughly stated as follows: Given a training set consisting of ordered pairs of feature vectors and

their associated labels (which might be discrete or continuous), the task of a learning algorithm is to learn a functional map from the feature space to label space. A learning algorithm is said to be more powerful if it is able to learn mappings such that it can generalize well and make correct predictions on test data-points on which it was not trained. Since the functional map under consideration might be highly non-linear, learning algorithms that output only a single mapping (frequently referred to as the hypothesis) might suffer from statistical, computational and representation issues that restrict them from learning good mappings. One way of solving this problem is to transform the feature space into a more suitable and "richer" representation such that learning using this new representation gives much better functional maps as compared to the original representation. This is the motivation behind deep learning methods which have caused a new wave of excitement in the machine community since 2006 [17]. Another way of solving this problem atleast partly, is by using ensemble learning methods [18],[19],[20]. The basic idea behind ensemble methods is that they involve running a "base learning algorithm" multiple times, each time with some change in the representation of the input (e.g. only considering a subset of features in each run) so that a number of diverse predictions (or maps) could be obtained. This diversity in prediction is then exploited to get better predictions. Thus ensemble methods approach the said problem by both trying to learn multiple functional maps and also by learning a more distributed and hence "richer" representation of the input space at the same time. In the next section we describe a method to use clustering for bootstrapping.

3.1 Clustering for Bootstrapping

In earlier work we introduced the idea of using clustering for bootstrapping [3], [4], [5], [6]. This idea was quite unlike other bagging methods which use a random subset to bootstrap. Thus, it had the potential advantage that the subsets used to bootstrap could be more interpretable. Before we generalize this methodology using co-clustering we first briefly describe the methodology using clustering.

The training set was first clustered into k disjoint clusters. A linear regression model was trained on each of the clusters only based on the training points assigned to that cluster. Since each such linear regression was a representative of only one cluster, we called it a cluster model. Thus, for a given k, there would be k cluster models. But since all the clusters are mutually exclusive, the training set is represented by all the cluster models taken together. This is called a prediction model (PM_k). For an incoming test point on which a prediction is to be made, we first identify the cluster that point belongs to. After the cluster has been identified, the appropriate cluster model could be used to make a prediction for that point. Now note that we don't specify the number of clusters in the above. Hence, we can change the granularity of the clustering from 1 to some high value, say K. In each instance we would get a different prediction model (a special case would be PM_1, which would basically be when one linear regression model is trained on the entire dataset). Thus, we would obtain a set of K prediction models each of which would make a separate prediction on the test set. Since we vary the granularity of the clustering, each of these predictions are different, this diversity in prediction could be used by averaging all the (or half) the predictions obtained to get a single much stronger prediction.

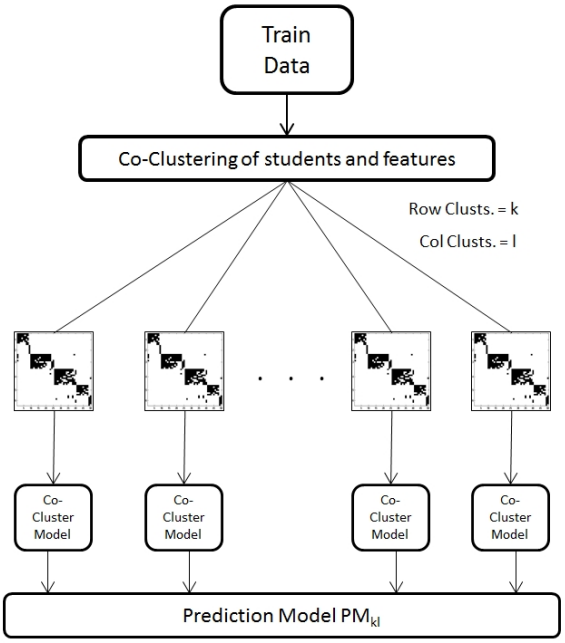

Figure 1: Finding a Prediction Model, PM_{kl} with k row clusters and l column clusters

3.2 Co-Clustering for Bootstrapping

Note that the clustering is only one-way. That is, bootstrapping is done by only changing the data instances available for each cluster model (by changing the number of cluster models itself) but the number of features used in each case is the same. A cluster basically is a bunch of rows in the data matrix with all columns. A co-cluster on the other hand would be a "block" in the data matrix with a sub-set of rows and a sub-set of columns assigned to each "co-cluster". Thus a co-clustering could be thought of as a simultaneous clustering and dimensionality reduction of the data. Note that a clustering is only a special case of co-clustering when the columns are not clustered at all (or have only one column cluster).

Clearly, the above bagging methodology can be suitably modified using co-clustering. For a given number of row clusters k and column clusters l we could have k co-clusters where-in each cluster has only some features assigned to it (note that the definition is symmetric i.e we could think of this as l co-clusters). For each co-cluster we train a separate linear regression model only using the data instances and features assigned to it. We thus obtain k Co-Cluster Models. Like in the above case for clustering, the combination of the k co-cluster models would be considered to be a Prediction Model which makes a single prediction on the test set. We can then vary k from 1 to some value K and l from 1 to some value L. By doing so, we would get a total of $K \times L$ prediction models. We then average a subset of the predictions made by these models to obtain a much stronger prediction.

There are some interesting aspects to such a methodology using co-clustering. For $k = 4$ and $l = 4$, the grid in Figure

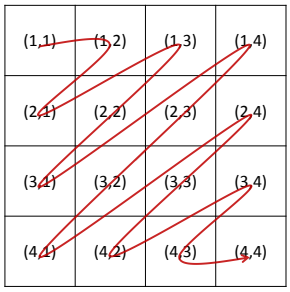

Figure 2: Ordering the Co-Cluster Prediction Models, PM_{kl}

2 illustrates all the Prediction Models (PM_{kl}) that could be obainted by co-clustering. The Prediction Model $PM_{1,1}$ represented by $(1, 1)$ is simply the case when there is one data cluster and only one feature cluster i.e the original data matrix itself. The prediction model for this case would simply be training a linear regression on the entire dataset, considering all the features. The first column of this grid represents the case when the number of feature clusters is just one, while the number of row clusters are changed. Note that this is simply the methodology described above in Section 3.1 using clustering. The first row of this grid is also equally interesting. In this case the number of row clusters is always one i.e the entire dataset is considered in all co-clusters, while the column clusters are successively changed. It should be noted that this is a sort of a step-wise regression, where a linear regression is trained on the entire dataset but the number of features that are used to train it are changed (usually reduced as l increases). All the other cases are a cross between these two extreme cases. We see that it seems plausible that a bagging strategy using co-clustering if averaged properly could definitely have more predictive power as it generates diversity by considering a different subset of data instances and features each time, consequently also generating a much larger set of predictions.

3.3 Blending Predictions

As mentioned before, the method for combining the predictions returned by the various prediction models is a naive averaging strategy. When the prediction models were generated by clustering (PM_k), we either averaged the first $K/2$ predictions (where K was the maximum number of clusters) [6] or we learned the best number of prediction models that could be averaged by an internal cross-validation [6]. The averaging idea is not immediately straightforward when co-clustering is used to generate the prediction models. This is because the prediction models are obtained by changing two parameters. It is also observed that prediction models with a high k or l return poor accuracies, thus it wouldn't be useful to average predictions from all the PM_{k1} models first and then PM_{k2} models and so on (i.e. traversing the grid row-wise or column-wise). Since high values of k and l are counter-productive, we take the order of the prediction models such that the sizes of k and l increase uniformly. This ordering is illustrated by the curve in Figure 2. The first half of this reordered set of predictions are then averaged.

4. EXPERIMENTAL VALIDATION

In this section we report experimental results for using co-clustering for bagging and compare results with the benchmark (PM_{11}) and clustering alone.

4.1 Dataset Description and Context

We primarily experiment with two datasets in this study. This data was collected to study if dynamic assessment, which has long been advocated as an effective method for assessment, was actually better than the traditional static assessment [21], [22]. Dynamic assessment is an interactive approach to student assessment which is primarily based on how much help a student requires during a practice test. Traditional static testing only takes into account the percentage of questions that the student gets correct. Feng *et al.* [23] showed that features that only recorded how much assistance a student got while interacting with a tutor alone were better predictors of student performance in post-tests held later in the year as compared to how many questions students got correct. This was confirmed in subsequent studies [4], [5]. Thus if Co-Clustering is able to improve predictions, then this study could further lend weight to the idea that dynamic testing is indeed better than static testing and that we could further improve upon PM_{11}. It must be noted that PM_{11} would correspond to results reported in [23] which were better than static assessment. PM_{11} basically corresponds to the condition when all the dynamic features are considered and all of the training set is used to train a predictor.

The datasets come from the 2004-05 and 2005-06 school years, the first two full years when ASSISTments.org was used in schools in Massachusetts. ASSISTments is an e-learning tutoring system developed at Worcester Polytechnic Institute which assesses students as it assists. These datasets contain features that measure the interaction of students with the tutor and their actual final grades, which they obtained at the end of the year in the Massachusetts state test (MCAS). There a total number of six features in these datasets **1) DA Original Count** is the number of questions that the students answered with assistance in the dynamic condition. **2) DA Original Percent Correct** is the percent of questions of feature 1 that students get correct . **3) DA Scaffold Percent Correct** is the percentage on tutorial help questions that students get correct. **4) DA Average Time** is the average time that a student spends on a question **5) DA Average Attempt** is the average number of attempts students made per question. **6) DA Average Hints** is the average number of hints that students used. The task is to use these interaction features to predict the MCAS scores that students might get at the end of the school year. The static condition feature is percentage of questions answered correct in static testing. This feature is never used for making predictions for the dynamic condition. The data in the 2004-05 set (ASSISTments 2004-05) is for 628 students, while the 2005-06 data (ASSISTments 2005-06) is for 761 students.

For experimentation we do a five fold cross-validation on the dataset and report results for the base condition (PM_{11}) and the various blended results which were obtained by averaging as discussed in Section 3.3. For the sake of comparison we also include results with k-means clustering too. In both cases we consider the ensembled results, with the top K predictions averaged as described in [4], [5] and also in Section 3.1. Following results in [4] and [5] we report results in terms of the mean absolute difference (MAD).

Finally, for pre-processing: As mentioned in Section 2, to

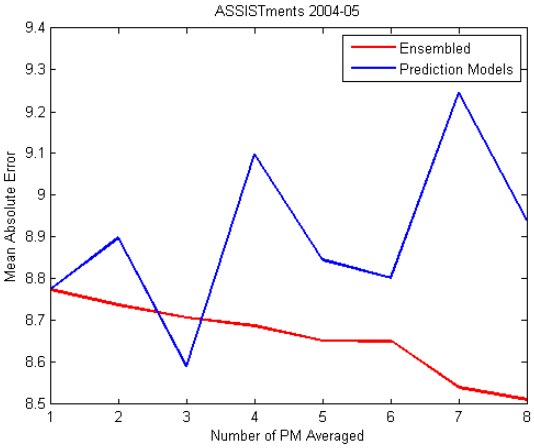

Figure 3: Performance on the 2004-05 Set

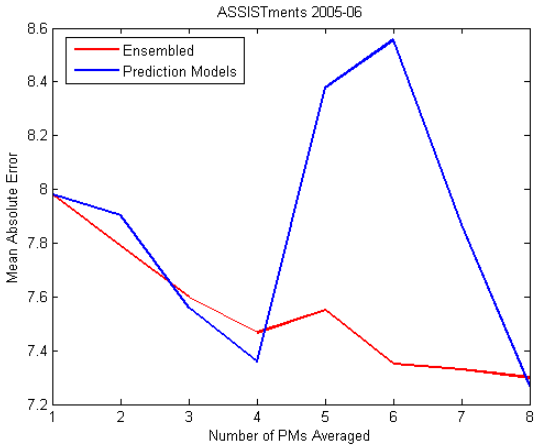

Figure 4: Performance on the 2005-06 Set

obtain a bipartite partitioning A must contain values that are either binary or scaled between 0 and 1. Thus, in each fold each feature column is scaled to between 0 and 1 so that A_n could be considered a co-occurrence matrix. This marks a slight difference from earlier papers in which the feature scaling was done so as to map all the data-points to between -1 and 1 by using the mapminmax command of MATLAB. This slight difference might result in a small variation in the results.

4.2 Experimental Results

We first report results on the ASSISTments 2004-05 dataset. The five fold cross-validated results using co-clustering are reported in Figure 3. The number of row clusters (k) and the number of column clusters (l) were restricted to 4 each. This resulted in 16 prediction models. The x-axis in the graph represents the first eight prediction models on doing co-clustering, while the y-axis simply gives the mean absolute error. We observe that the accuracy of co-clustering alone is quite bad (as seen by the blue line) as compared to the baseline (PM_{kl}, which is basically the result for x = 1 in this graph. Note that the baseline is the dynamic condition of Feng [23]). These predictions are those given by the first elements of the ordered set of co-cluster prediction models as defined in Section 3.3. However, averaging these prediction models successively gives better and better predictions (as can be seen by the red line).

Similar results were reported in the ASSISTments 2005-06 dataset as shown in Figure 4. In this dataset the prediction models are far worse than the ensembled results as compared to the previous dataset. Again, we obtain 16 prediction models after co-clustering and successively average the first eight (the first with second, the first with second and third and so on) after they have been arranged in the way suggested in Section 3.3. Again the ensembled results do much better over the baseline (we report exact figures and significance in Tables 1 and 2).

In Table 1 we compare the mean absolute errors when predictions of the first five prediction models are bagged. We report results when the Prediction Models are obtained both by using co-clustering and using k-means clustering on the ASSISTments 2004-05 dataset. The figures in bold indicate statistical significance over the baseline prediction on a paired t-test. Results in Table 2 compare the predictions obtained by using co-clustering and k-means for bagging on the ASSISTments 2005-06 dataset.

Table 1: Comparison of predictions based on k-means and Co-Clustering for the ASSISTments 2004-05 Dataset. Figures in bold indicate significance over the baseline on paired t-test. Numbers are Mean Absolute Errors. Also note that Pred. Model 1 corresponds to the baseline

Pred. Models	Co-Clust	k-means
1	8.7741	8.7741
2	8.7379	8.7518
3	**8.7087**	**8.6725**
4	**8.6879**	8.7153
5	**8.6574**	**8.7100**

The results are significantly better over the baseline and also indicate that the dynamic assessment condition returns a much better prediction of student test scores as compared to the static condition. It has already been noted that the static test condition results are significantly worse as compared to even the baseline by [23] and [4], and thus we don't report results for the static condition.

5. DISCUSSION AND FUTURE WORK

The datasets that were used for the validation of this bagging technique, which is based on co-clustering were not very large and did not have a large number of columns. Thus,

Table 2: Comparison of predictions based on k-means and Co-Clustering for the ASSISTments 2005-06 Dataset. Figures in bold indicate significance over the baseline on paired t-test. Numbers are Mean Absolute Errors. Also note that Pred. Model 1 corresponds to the baseline

Pred. Models	Co-Clust	k-means
1	7.9822	7.9822
2	**7.7716**	**7.8185**
3	**7.5990**	**7.8034**
4	**7.4680**	**7.7815**
5	**7.5503**	**7.6487**

these results were initially surprising. One would imagine that in a dataset which has a small number of features, perhaps a feature selection might not be too helpful. However, our experiments show us otherwise. The results that we obtain, while modest improvements show that this technique though simple can give access to a novel source of variance in the data. It can potentially also have some nice properties in terms of returning simpler and more interpretable groups. For example, it was earlier pointed out that one row of the prediction models were actually nearly like a linear regression model in which the features are successively eliminated. At the same time it was observed that one column of the prediction models were actually just the various prediction models that we obtained on clustering alone as reported in some previous work. It would be interesting to see how the Co-Clusters (which are basically blocks in the data matrix) on a student-item dataset would pair clusters of student proficiency to clusters of item performance which could be seen as a sort of a subject treatment interaction.

In the literature, it has been said that the real strength of co-clustering is with binary valued data, co-occurrence tables and basically in scenarios which involve collaborative filtering. Hence, datasets which are basically a student by item matrix would be an ideal candidate for trying out this technique. In the KDD Cup 2010 Töscher and Jahrer modelled student response data as a collaborative filtering task and used matrix factorization techniques for the same. Given the connections of co-clustering with matrix factorization, it is worth investigating how useful it could be in such a setting.

In [3], the authors clustered students based on tutor interaction features and then trained separate Knowledge Tracing models for students based on the cluster they were in. This was done so because it was not possible to cluster the item sequences directly and an indirect approach had to be taken. This co-clustering technique seems to give an alternative by which such matrices might be clustered more readily without the need to cluster the tutor interaction features.

In summary, in this paper we propose a bagging technique that uses co-clustering and demonstrate that it's performance is better than that obtained by bagging using clustering. We also suggest that it is most suitable for datasets which are like co-occurrence tables and believe that it would be a good direction for future work since such student-item datasets are usually of this form.

6. ACKNOWLEDGEMENTS

The authors are indebted to all the funders listed over here `http://www.webcitation.org/5xp605MwY`. The research of the third author (Gábor Sárközy) is supported in part by the National Science Foundation under Grant No. DMS-0968699.

7. REFERENCES

[1] Felder, R.M., and Brent. R.,Understanding Student Differences, Journal of Engineering Education, Vol. 94, No. 1, 2005, pp. 57-72.

[2] Bransford, J.D., Brown, A.L., and Cocking, R., eds., How People Learn: Brain, Mind, Experience, and School, Washington, D.C.: National Academy Press, 2000.

[3] Pardos, Z. A., Trivedi, S., Heffernan. N. T., and Sárközy. G. N., Clustered Knowledge Tracing, In the Proceedings of the 11th International Conference on Intelligent Tutoring Systems 2012, Chania, Greece.

[4] Trivedi, S., Pardos, Z. A., Heffernan, N. T., Clustering Students to Generate an Ensemble to Improve Standard Test Score Predictions, G. Biswas et al. (Eds.): AIED 2011, LNAI 6738, In The proceedings of the 15th International Conference on Artificial Intelligence in Education 2011, Auckland, New Zealand, pp. 377-384.

[5] Trivedi, S., Pardos, Z. A., Sárközy, G. N., Heffernan, N. T., Spectral Clustering in Educational Data Mining, In Proceedings of the 4th International Conference on Educational Data Mining 2011, Eindhoven Netherlands, pp. 129-138.

[6] Trivedi, S., Pardos, Z. A., and Heffernan, N. T., The Utility of Clustering in Prediction Tasks, IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics (under review).

[7] Hartigan, J. A., Wong, M. A., Algorithm AS 136: A K-Means Clustering Algorithm, Journal of the Royal Statistical Society, Series C (Applied Statistics), 1979, 28 (1): pp. 100-108.

[8] Hartigan, J. A., Direct Clustering of a Data Matrix, Journal of the American Statistical Association, 67(337): pp. 123-129, 1972.

[9] Dhillon, I. S., S, Mallela., and Kumar, R., A divisive Information-Theoretic Feature Clustering Algorithm for text classication. Journal of Machine Learning Research, 3(4): pp. 1265-1287, 2003.

[10] Dhillon, I. S., Co-clustering documents and words using bipartite spectral graph partitioning. In Proceedings of the 7th International Conference on Knowledge Discovery and Data Mining, pp. 269-274, 2001.

[11] Hanisch, D., Zein, A., Zimmer, R., Lengauer, T., Co-clustering of biological networks and gene expression data. Bioinformatics 18 (Suppl.), 2002, pp. 145-154.

[12] D'Haeseleer, P., Liang, S., and Somoyogi, R., Genetic Network Inference: From co-expression clustering to reverse engineering, Bioinformatics, 16, 2000, pp. 707-726.

[13] George, T., and Merugu, S., A scalable collaborative ltering framework based on co-clustering. In Proceedings of the IEEE Conference on Data Mining, pp. 625-628, 2005.

[14] Dhillon, I., Mallela, S., and Modha, D., Information-theoretic co-clustering. In Proceedings of the 9th International Conference on Knowledge Discovery and Data Mining (KDD), pp. 89-98, 2003.

[15] Banerjee, A., Dhillon, I. S., Ghosh, J., Merugu, S., and Modha, D. S, A Generalized Maximum Entropy Approach to Bregman Co-clustering and Matrix Approximation, Journal of Machine Learning Research, 8, pp. 1919-1198, 2007.

[16] J. Shi, and J. Malik, Normalized Cuts and Image Segmentation, IEEE Transactions on Pattern Analysis and Machine Intelligence, 22 (8), pp. 888-905, 2000.

[17] Bengio, Y., Learning deep architectures for AI. Foundations and Trends in Machine Learning, 2(1): pp. 1-127, Now Publishers, 2009.

[18] Dietterich, T.G., Ensemble Methods in Machine Learning. In: Kittler, J., Roli, F. (eds.) First International workshop on Multiple Classifier Systems. LNCS, pp. 1-15. Springer, New York, 2000.

[19] Dietterich, T.G., An Experimental Comparison of Three Methods for Constructing Ensembles of Decision Trees: Bagging, Boosting, and Randomization, Machine Learning 40, pp. 139-157, 2000.

[20] Breiman, L., Random Forests, Machine Learning 45(1), pp. 5-32, 2001.

[21] Grigerenko, E.L., Steinberg, R.J.: Dynamic Testing. Psychological Bulletin 124, pp. 75-111, 1998.

[22] Campione, J. C., Brown, A. L.: Dynamic Assessment: One Approach and some Initial Data. Technical Report. No. 361. Cambridge, MA. Illinois University, Urbana, Center for the Study of Reading. ED 269735, 1985.

[23] Feng, M., Heffernan, N.T., Koedinger, K.R.: Addressing the assessment challenge in an online system that tutors as it assesses. User Modeling and User-Adapted Interaction: The Journal of Personalization Research 19(3), 2009.

[24] Mohar, B., The Laplacian spectrum of graphs. In Graph theory, combinatorics, and applications. Vol. 2 (Kalamazoo, MI, 1988), New York: Wiley, 1991, pp. 871-898.