A Framework for Simultaneous Co-clustering and Learning from Complex Data

Meghana Deodhar Joydeep Ghosh

{deodhar/ghosh}@ece.utexas.edu

IDEAL-2007-08*

Intelligent Data Exploration & Analysis Laboratory

(Web: http://www.ideal.ece.utexas.edu/)

Department of Electrical and Computer Engineering
The University of Texas at Austin
Austin, Texas 78712
U.S.A.

March 28, 2007

Abstract

For difficult classification or regression problems, practitioners often segment the data into relatively homogenous groups and then build a model for each group. This two-step procedure usually results in simpler, more interpretable and actionable models without any loss in accuracy. We consider problems such as predicting customer behavior across products, where the independent variables can be naturally partitioned into two groups. A pivoting operation can now result in the dependent variable to show up as entries in a "customer by product" data matrix. We present a model-based co-clustering (meta)-algorithm that interleaves clustering and construction of prediction models to iteratively improve both cluster assignment and fit of the models. This algorithm provably converges to a local minimum of a suitable cost function. The framework not only generalizes co-clustering and collaborative filtering to model-based co-clustering, but can also be viewed as simultaneous co-segment-ation and classification or regression, which is better than independently clustering the data first and then building models. Moreover, it applies to a wide range of bi-modal or multimodal data, and can be easily specialized to address classification and regression problems. We demonstrate the effectiveness of our approach on both these problems through experimentation on real and synthetic data.

1 Introduction

While it is common practice to develop a single learned model (e.g., a classification or regression model, or a single ensemble of multiple base-learners) to characterize a given dataset, for many problems it is practically advantageous to partition the population into multiple, relatively homogeneous segments and then develop separate models for each segment [BG93, DBSP93, OH01, LS00]. For example, an e-tailor may attract different types of browsers, from casual shoppers to bulk purchasers, and one may want to model their purchasing inclinations separately. Similarly while forecasting electric load usage, it is advisable to building separate predictive models for weekdays, weekends and holidays. Advantages of such divide-and-conquer approaches include not only improved accuracy and reliability in general, but also improved interpretability as well, since the component models are often far simpler [Sha96]. For example, the mixtures-of-experts model [JJ94] typically uses linear regression models composed together to form non-linear maps. Note that this model achieves (soft) partitioning and learning simultaneously; more common are situations such as the load forecasting application, where the partitioning is done apriori based on domain knowledge or a separate segmentation routine [BG93], [DBSP93].

This paper is concerned with situations where the independent variables can be naturally partitioned into two (or more) groups that are associated with their corresponding modes. We then simultaneously cluster along each mode, as well as fit a learned model to each co-cluster. The approach can *alternatively be viewed as a model-based generalization of biclustering or co-clustering*, which is a technique that simultaneously clusters along multiple axes and has been successfully applied in several domains like text clustering and microarray data analysis [CDGS04], [CC00], [DMM03]. Co-clustering, is traditionally applied to a matrix of data values, where the rows are data points and the columns are features, e.g. in microarray data the rows are genes and columns are experiments, in recommender systems the rows are customers and the columns are products. Co-clustering exploits the duality between the two axes to improve on single-sided clustering. In the alternative viewpoint, along with the data matrix, a set of variables is associated with the rows, and another set with the columns. For each co-cluster, a model is learned to predict a matrix cell value given the corresponding row and column attributes. Generalization refers to predicting the missing values in the data matrix, as well as values when new rows/columns are added.

Example: To concretize the above discussion, consider the problem of predicting customer purchase decisions (recommending products to customers). The dataset in this case is a matrix of customers by products, where the cell values are class labels representing whether a customer buys a certain product or not. This matrix will have missing values where the corresponding customer-product choice is unknown. Each customer is described by a set of attributes, for example, demographics and each product also has attributes, which could include the price, market share, quality, etc. The problem is to predict the choices for the missing customer-product combinations, as well as behavior of new customers, or choices made for new products. Note that collaborative filtering approaches for this problem will make use only of the matrix entries and ignore customer/product attributes [HKBR99, GM05]. On the other extreme, a typical classification model will form a map between the feature vector for a given customer-product pair and the corresponding matrix entry, but will not consider nearby customers or products in this process. For the classifier, the dependent variable values are nothing but the matrix entries, and the independent variables are grouped into variables associated with the

rows and variables associated with the columns. For a diverse population of customers and a wide range of different products, it is unlikely that all the customers have the same choice models for all the products. It is more natural for a subset of the customers to have similar choice models across a subset of products.

A co-clustering approach will simultaneously cluster the customers and products based on the matrix entries. It will then use the entries of the corresponding co-cluster to predict a missing value. If done properly, this gives better results than standard recommender systems [GM05], however this approach still ignores the customer and product attributes, i.e., the prediction is solely based on the value of the dependent variable in a suitably identified neighborhood. Our approach exploits both such neighborhood information as well as the available customer/product attributes. The idea is to co-cluster the entire data matrix into blocks of customers and products such that each block can be well characterized by a single predictive model. Note that the similarity between two data entries is now determined not by the similarity between the values themselves, but rather between the corresponding predictive models. Moreover, our model based co-clustering-cum-learning algorithm achieves this by interleaving clustering and construction of classification models to iteratively improve both cluster assignment and fit of the models. This simultaneous approach is better than independently clustering the data first and then building classification models. We also exhibit a cost function that is steadily decreased in both steps of the iterative process, till one reaches a local minimum, thereby guaranteeing convergence.

The above approach is not restricted to classification. To highlight this, we also consider the problem of building multiple regression models, one for each co-cluster, for the application of predicting the number of items of a given product purchased by a customer, using a formidable, real data set in Section 8.2. The overall approach remains the same, only the specifics of the learned models and of the measure of fit changes.

In the rest of the paper we refer to the data points as customers and axes as products, based on our motivating applications. Our approach is however not restricted to a customer-product matrix, and is applicable to any bi-modal data set. It can also be extended to multi-modal data, e.g. a 3-D tensor (data cube) with sets of variables associated with one or more of the axes.

Notation: Small letters represent scalars e.g. a,z, small, bold face letters represent vectors e.g. $\mathbf{b}, \mathbf{c}, \boldsymbol{\beta}$, capital letters like Z, W represent matrices. Individual elements of a matrix e.g. Z are represented as z_{ij} where i and j are the row and column indices respectively.

2 Related Work

There are several examples of the use of localized prediction models in load forecasting systems, where clustering is used to distinguish smaller, homogenous groups of data and a prediction model is then fitted for each cluster in a two step sequential process [BG93], [DBSP93]. Clustering based prediction models have also been widely used in economics [OH01], [LS00]. Sfetsos *et al.* propose an iterative algorithm to cluster time series data such that each cluster consists of data points with similar linear models [SS04]. This is followed by a heuristic to identify a single cluster to be used for future predictions. This clustering algorithm is one sided and linear model based, a special case of our co-clustering algorithm.

In the bioinformatics domain, clustering of genes is often used as a preprocessing step for the

classification of experiments (samples) in microarray data analysis [LKM05], [CMHD03], [JY03]. A cluster is represented by the mean of the expression profiles across all its member genes, which acts as a dimensionality reduction step for the classification process. This helps to reduce gene redundancies and constructs parsimonious and more interpretable classification models. A simultaneous clustering and classification algorithm is proposed by Zhang *et al.* [ZNB05], which uses a voting based classifier ensemble to improve a clustering solution. The labels assigned by an initial clustering are used to train a set of diverse classifiers. Data points that lie on cluster boundaries are relabeled by combining the classifier predictions using a majority vote. This process is iterated to refine the clustering solution.

Co-clustering (also known as biclustering) has been used in several diverse data mining applications like clustering microarray data [CC00], [CDGS04], text mining [DMM03] and marketing applications. Most clustering or co-clustering approaches cannot handle missing data and assume a full data matrix. However the formulation in [BDG$^+$06] readily handles missing data, and has been shown to perform better than traditional collaborative filtering techniques in a recommender system setting [GM05], where the data is a matrix of customer-movie ratings. The known ratings are used to simultaneously cluster customers and movies and compute summary statistics for the co-clusters, which are then used to predict unknown ratings, using an instance of the Bregman co-clustering algorithm [BDG$^+$06]. Recently Ying *et al.* proposed a recommendation model that emphasizes the importance of using missing recommendation ratings as part of the model rather than ignoring them completely [YFW06]. They demonstrate that by jointly modeling whether and how an item was rated (selection and rating respectively), the accuracy of existing recommendation systems can be substantially improved.

The idea of simultaneous clustering and regression was introduced in the marketing literature in the paper by Wedel and Steenkamp, who proposed a generalized fuzzy clusterwise regression technique to find both customer segments and market structure [WS91]. Each cluster includes fractional membership from all customers and products and is hence a fuzzy co-cluster. Each cluster has a regression model that predicts the preferences as a linear combination of the product attributes. The cluster memberships and models are estimated so as to reduce the total squared error between the actual preferences and the predicted preferences. But no one has attempted simultaneous co-clustering and predictive modeling.

3 Problem Definition

We now describe the problem formulation for the classification setting. Let m be the total number of customers and n the total number of products. The data can be represented as an $m \times n$ matrix Z of customers and products, with cells z_{ij} representing the corresponding class labels, e.g. whether customer i buys product j or not. Throughout the following discussion we assume that we are dealing with a 2 class problem and $z_{ij} \in \{-1, +1\}$, however the algorithm can easily be generalized to deal with multiclass settings. The problem formulation and solution for regression models is given in Section 5. A weight w_{ij} is associated with each cell z_{ij}. The weights of the known (training) matrix cell values are set to 1. The missing cell values, that are to be predicted are given a weight of 0. In general, the weight is not restricted to 0 or 1 and can take values between 0 and 1. This formulation allows the prediction framework to deal with data uncertainties, where less certain values can be given lower but non-negative weights.

A customer i has attributes $\mathbf{C_i}$, and product j has attributes $\mathbf{P_j}$. It is assumed that each class label z_{ij} is primarily determined by the attributes of the corresponding customer-product pair and is generated by a certain model involving these attributes. We assume this model to be a logistic regression model [1] where the log odds is modeled as a linear combination of the customer and product attributes given by

$$ln\frac{P(z_{ij}=1|\mathbf{x_{ij}})}{1-P(z_{ij}=1|\mathbf{x_{ij}})} = f(\mathbf{x_{ij}}),$$

where $\mathbf{x_{ij}}^T = [1, \mathbf{C_i}^T, \mathbf{P_j}^T]$ is a vector consisting of the customer and product attributes, and $f(\mathbf{x_{ij}}) = \boldsymbol{\beta}^T x_{ij}$ is a linear model with parameters $\boldsymbol{\beta}^T = [\beta_0, \boldsymbol{\beta_c}^T, \boldsymbol{\beta_p}^T]$. The similarity of the cell values is now defined based on the similarity of their underlying logistic regression models. The aim is to simultaneously cluster the rows (customers) and columns (products) into a grid of k row clusters and l column clusters [2], such that the class labels within each co-cluster are predicted by a single common classification model. The co-cluster assignments along with the classification models for the co-clusters can be used to predict the class labels for missing customer-product combinations.

Formally, let ρ be a mapping from the m rows to the k row clusters and γ be a mapping from the n columns to the l column clusters. We want to find a co-clustering defined by (ρ, γ) and associated set of classification models $\{\boldsymbol{\beta^{gh}}\}$ that minimize the following objective function

$$\sum_{g=1}^{k}\sum_{h=1}^{l}\sum_{u:\rho(u)=g}\sum_{v:\gamma(v)=h} w_{uv} ln(1+exp(-z_{uv}\boldsymbol{\beta^{gh}}^T\mathbf{x_{uv}})), \tag{1}$$

where z_{uv} is the original value (class label) in row u, column v of the matrix, with associated weight w_{uv}. Here $\boldsymbol{\beta^{gh}}$ denotes the vector of coefficients of the model associated with the co-cluster that the cell value z_{uv} is assigned to. Since the weights for the missing z_{uv} values are 0, the objective function essentially ignores them and is simply the log loss summed over all the known elements of matrix Z. Minimizing this objective function is equivalent to maximizing the log-likelihood of the data.

4 Simultaneous Co-clustering and Classification

A co-clustering (ρ, γ), that reduces the cost function (1) can be obtained by a simple iterative algorithm. Since the objective function is the log loss summed over all the elements of the matrix, it can be expressed as a sum of row or column losses. If row u is assigned to row cluster g (i.e. $\rho(u) = g$), the row error is

$$E_u(g) = \sum_{h=1}^{l}\sum_{v:\gamma(v)=h} w_{uv} ln(1+exp(-z_{uv}\boldsymbol{\beta^{gh}}^T\mathbf{x_{uv}})).$$

Since any missing values in the row u will have a weight 0, the error $E_u(g)$ is effectively computed only over the known values in row u. For a given column clustering and model

[1] The focus of this paper is on simultaneous co-clustering and classification, and not obtaining the best possible classifier, therefore we have chosen a standard, fairly flexible classifier rather than experiment with a multitude of classifier options.

[2] This form of co-clustering is often called partitional co-clustering [MO04]

parameter sets $\{\boldsymbol{\beta^{gh}}\}$, the best choice of the row cluster assignment for row u is the g that minimizes this error, that is,

$$\rho^{new}(u) = argmin_g E_u(g).$$

Each row is hence assigned to the row cluster that minimizes the row error. A similar approach is used to (re)-assign columns to column clusters. Such row and column cluster updates hence decrease the objective function and improve the clustering solution. Note that updating column cluster assignments could cause the best row assignments to change and vice versa. Thus optionally, the row and column cluster reassignment steps can be repeated several times and in arbitrary order until both row and column cluster memberships converge.

Given the current row and column cluster assignments, the co-cluster models need to be updated, i.e. the co-efficient vector $\boldsymbol{\beta}$ has to be updated for each co-cluster. To update the model for a row cluster g of size r and column cluster h of size c, train a logistic regression model with the $r*c$ values within the co-cluster, weighted by their corresponding weight values. The missing values present in the co-cluster have weights of 0 and are essentially ignored. The logistic regression model is hence trained using only the known training samples $(\mathbf{x_{ij}}, z_{ij})$. In the more general case of arbitrary valued weights, this step involves updating a weighted logistic regression model [LL03] rather than a simple logistic regression model. The output will be an updated vector $\boldsymbol{\beta^{gh}}$ of coefficients that minimizes the model log loss given by

$$L = \sum_{u=1}^{r} \sum_{v=1}^{c} w_{uv} ln(1 + exp(-z_{uv} \boldsymbol{\beta^{gh}}^T \mathbf{x_{uv}})).$$

The model update step is hence guaranteed to decrease the objective function.

The resulting algorithm is a simple iterative algorithm described in Figure 1. Step 1 minimizes the objective function due to the property of logistic regression, steps 2(a) and 2(b) directly minimize the objective function. The objective function hence decreases at every iteration. Since this function is bounded from below by zero, the algorithm is guaranteed to converge to a local minimum. This algorithm could also be extended to classification models other than logistic regression. In this case the loss function and the update models step will be modified, but the overall approach will still be the same.

Predicting missing class labels: After the co-cluster assignments and the co-clusterwise classification models are obtained by the algorithm, the missing class labels can be predicted easily. Let z_{uv} be a missing cell value that has been assigned to row cluster g and column cluster h. $\mathbf{x_{uv}}$ is the vector of attributes of row u and column v and $\boldsymbol{\beta^{gh}}$ represents the model parameters of the logistic regression model of the assigned co-cluster. The logistic regression model is used to obtain the probability of z_{uv} of belonging to the positive class as follows

$$P(z_{uv} = 1) = \frac{1}{1 + e^{-\boldsymbol{\beta^{gh}}^T \mathbf{x_{uv}}}}$$

A suitable threshold t is used to convert the probabilities into class labels i.e. $z_{uv} = 1$ if $P(z_{uv} = 1) > t, z_{uv} = -1$ otherwise.

5 Simultaneous Co-clustering and Regression

In the regression setting, Z is an $m \times n$ matrix of "customers" and "products", with cells representing the corresponding customer-product preference values, ratings or choice probabilities.

Algorithm
Input: $Z_{m\times n}$, $W_{m\times n}$, $C = [\mathbf{C_1}..\mathbf{C_m}]$, $P = [\mathbf{P_1}..\mathbf{P_n}]$
Output: Co-clustering (ρ, γ) and co-cluster models $\boldsymbol{\beta}$'s
1. Begin with a random co-clustering (ρ, γ)
2. Repeat

Step 1
3. Update co-cluster models
4. for $g = 1$ to k do
5. for $h = 1$ to l do
6. Train a logistic regression model with all the training samples $(\mathbf{x_{ij}}, z_{ij})$ in
8. co-cluster (g, h), with associated weights w_{ij}, to obtain an updated $\boldsymbol{\beta^{gh}}$.
10. end for
11. end for

Step 2(a)
12. Update ρ - assign each row to the row cluster that minimizes the row error
14. for $u = 1$ to m do
15. $\rho(u) = argmin_g \sum_{h=1}^{l} \sum_{v:\gamma(v)=h} w_{uv} ln(1 + exp(-z_{uv} \boldsymbol{\beta^{gh}}^T \mathbf{x_{uv}}))$
17. end for

Step 2(b)
18. Update γ - assign each column to the column cluster that minimizes the column error
20. for $v = 1$ to n do
21. $\gamma(v) = argmin_h \sum_{g=1}^{k} \sum_{u:\rho(u)=g} w_{uv} ln(1 + exp(-z_{uv} \boldsymbol{\beta^{gh}}^T \mathbf{x_{uv}}))$
23. end for
23a. Optional: repeat steps 2(a) and 2(b) until convergence

until convergence
24. return (ρ, γ) and $\boldsymbol{\beta}$'s

Figure 1: Pseudo-code for simultaneous co-clustering and classification

Here we assume the generative model to be a linear model, where the preference value $z_{ij} \in \mathbf{R}$ is modeled as a linear combination of the corresponding customer and product attributes. The preference value is estimated as $\hat{z_{ij}} = \beta_0 + \boldsymbol{\beta_c}^T \mathbf{C_i} + \boldsymbol{\beta_p}^T \mathbf{P_j}$. Similar to the problem definition in section 3, the aim is to simultaneously cluster the customers and products into a grid of k row clusters and l column clusters, such that preference values within each co-cluster have similar linear models and can be represented by a single common model. We want to find a co-clustering defined by (ρ, γ) and the associated $k * l$ regression models that minimize the

following objective function

$$\sum_{g=1}^{k}\sum_{h=1}^{l}\sum_{u:\rho(u)=g}\sum_{v:\gamma(v)=h} w_{uv}(z_{uv}-\hat{z_{uv}})^2. \tag{2}$$

where $\hat{z_{uv}} = (\boldsymbol{\beta^{gh}})^T\mathbf{x_{uv}}$. The only difference here as compared to the classification case is the loss function, which is now squared loss rather than log loss. A co-clustering (ρ, γ), that minimizes the objective function can be obtained by an algorithm similar to the one described in section 4. The cluster reassignment steps assign each row or column to the row or column cluster that minimizes the row or column error.

The model for row cluster g of size r and column cluster h of size c is updated by solving

$$\min\|\mathbf{w}_{(\mathbf{r*c})\times\mathbf{1}}\mathbf{z}_{(\mathbf{r*c})\times\mathbf{1}} - X_{(r*c)\times p}\boldsymbol{\beta}_{p\times 1}\|_2^2,$$

where $\mathbf{z}$ is a vector of all the $r * c$ preference values in the co-cluster and $\mathbf{w}$ is a vector of the corresponding weights. X is the matrix of the corresponding row and column attributes. p is the total number of co-efficients to be estimated, including the intercept. In case of 0/1 weights, this is equivalent to ignoring missing values and updating the $\boldsymbol{\beta}$ for each co-cluster by least squares regression using only the non-missing (training) values within the co-cluster. In case of a general set of weights, the $\boldsymbol{\beta}$ is a solution to a weighted least squares problem. Least squares regression finds a solution for $\boldsymbol{\beta}$ that minimizes the sum of the squared errors between the original values and the predicted values. The model update step is hence guaranteed to decrease the objective function.

After the algorithm converges, the co-cluster assignments and co-clusterwise regression models can be used to predict unknown customer-product preference values. A missing value z_{uv} is predicted as $\hat{z_{uv}} = \boldsymbol{\beta^{gh}}^T\mathbf{x_{uv}}$.

6 Reduced Parameter Approach

The simultaneous co-clustering and prediction approach constructs $k \times l$ independent models, one per co-cluster, requiring a total of $(1 + |C| + |P|) \times kl$ parameters, where $|C|$ and $|P|$ are the number of customer and product attributes respectively. This model will have many parameters for large values of k and l and may overfit in cases where training data is limited. The other extreme is a single prediction model for all the data, with $(1+|C|+|P|)$ parameters, which might not be adequate. We propose an intermediate approach, which constructs $k \times l$ models but with smoothing or regularization achieved by sharing parameters across certain sets of models. The co-cluster models are constructed in such a way that the customer coefficients of all models for the same row cluster and the product coefficients of all models for the same column cluster are constrained to be identical. This reduced setting has $(1+|C|)\times k+(1+|P|)\times l$ parameters, which will be considerably lower than $(1 + |C| + |P|) \times kl$ for large k and l. We now describe how the model parameters can be obtained for the regression problem.

If z_{uv} is the original value in row u, column v of the matrix Z that is assigned to row cluster g and column cluster h, the predicted value $\hat{z_{uv}}$ is now given by

$$\hat{z_{uv}} = \beta^g_{c0} + \boldsymbol{\beta^{g}_{c}}^T\mathbf{C_u} + \beta^h_{p0} + \boldsymbol{\beta^{h}_{p}}^T\mathbf{P_v},$$

where β^g_{c0} and β^h_{p0} are the customer and product intercepts and $\boldsymbol{\beta^g_c}$ and $\boldsymbol{\beta^h_p}$ are the customer and product coefficient vectors for the row cluster g and column cluster h respectively.

We still want to find a co-clustering defined by (ρ, γ) and associated regression models that minimize the objective function (2). The row and column cluster assignment steps remain the same. The update models step now involves solving a constrained optimization problem. Instead of updating $k \times l$ linear models independently, this step now updates the k row cluster models, such that the product coefficients are fixed, and the customer coefficients are updated, and then the l column cluster models, in which the customer coefficients are fixed and the product coefficients are updated.

To update the model for row cluster g with r rows and n columns solve

$$min\|\mathbf{wy} - X_c[\beta^g_{c0}, \boldsymbol{\beta^g_c}^T]^T\|^2_2,$$

where X_c is an $(r * n) \times (1 + |C|)$ matrix of the customer attributes, with the first column set to 1 for the intercept. The response variable $\mathbf{y}$ is a vector with $r * n$ elements, given by

$$\mathbf{y} = \mathbf{z} - X_p[\beta^h_{p0}, \boldsymbol{\beta^h_p}^T]^T,$$

where $\mathbf{z}$ is a vector of all the $r*n$ preference values in the row cluster g with associated weights $\mathbf{w}$, $[\beta^h_{p0}, \boldsymbol{\beta^h_p}^T]$ is a vector of the product coefficients of the corresponding column clusters and X_p is a matrix of size $(r*n)\times(1+|P|)$ representing the product attributes corresponding to the preference values. The column cluster models are updated similarly. This update ensures that all the customer coefficients of models within the same row cluster and the product coefficients of models within the same column cluster are updated simultaneously and are identical.

7 Experimental Evaluation of Classification Results

7.1 Synthetic Datasets

The algorithm described in section 4 was first evaluated on a number of synthetic datasets. We used synthetic data for experimentation as an initial sanity check before working with real data. These experiments also indicate the amount of improvement localized classification models provide when the model assumptions match the generative model for the data. The synthetic dataset is created by generating $k \times l$ blocks of data, each corresponding to a true cluster of class labels generated by a logistic regression model. Each block has a different logistic regression model, with its coefficient vector $(\boldsymbol{\beta})$, consisting of the intercept and the coefficients for the customer and product attributes, set randomly. The blocks are then appended together in the form of a grid to form a data matrix, whose rows and columns are then randomly shuffled. To assign a class label to a cell z_{ij} within each block, we begin by taking a linear combination of randomly generated customer and product attributes with the co-efficient vector $y_{ij} = \boldsymbol{\beta^T x_{ij}}$. We then add random gaussian noise with variance σ^2 to all the y values. We obtain the probability of a cell belonging to the positive class as $P(z_{ij} = 1) = \frac{1}{1+e^{-y_{ij}}}$. A threshold of 0.5 is used to convert the probabilities into class labels ($z_{ij} = 1$ if $P(z_{ij} = 1) > 0.5, z_{ij} = -1$ if $P(z_{ij} = 1) <= 0.5$).

Table 1 describes the synthetic datasets that were used for experimentation. Dataset 1 and 2 are very similar, dataset 2 has more noise and hence a weaker relationship with the underlying generative model as compared to dataset 1. Dataset 3 and 4 are larger, also with a substantial

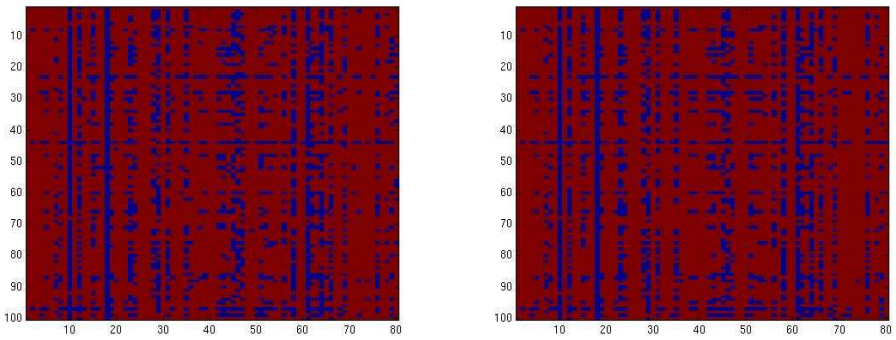
amount of noise. The datasets along with details of their generative models can be accessed at `http://www.ece.utexas.edu/~deodhar/modelCCData`.

	m,n	$\|C\|$,$\|P\|$	k,l	noise σ^2
Dataset 1	100, 80	3,4	3,2	5
Dataset 2	100, 80	3,4	3,2	15
Dataset 3	500, 300	3,4	4,3	5
Dataset 4	900, 500	5,5	8,6	5

Table 1: Synthetic Datasets

7.1.1 Results on Synthetic Data

Figure 2 displays the ability of the algorithm to reconstruct the original data matrix on synthetic dataset 2. The original data matrix is compared to the approximated matrix, where the class labels are obtained by the logistic regression models within each co-cluster. The red and blue cells represent positive and negative class labels respectively. One can observe that the reconstruction is quite close to the original.



(a) Original Data Matrix (b) Reconstructed Data Matrix

Figure 2: Reconstruction ability of models

Mutual information, which is a symmetric measure that quantifies the statistical information shared between distributions (Cover and Thomas, 1991) can be used to provide an indication of the information shared between the assigned cluster labels and the true labels. Let X and Y be two random variables that represent the assigned labels and the true labels respectively. Let $I(X,Y)$ denote the mutual information between X and Y. $I(X,Y)$ can be normalized so that it lies between 0 and 1 and allows easier interpretation and comparison. If $H(X)$ denotes the entropy of X, the normalized mutual information (NMI) [SG02] between X and Y is computed as:

$$NMI(X,Y) = \frac{I(X,Y)}{\sqrt{H(X)H(Y)}}$$

NMI can hence be used as a measure for evaluating a clustering result. The Normalized Mutual

Information (NMI) of the row and column cluster labels with the true labels is computed on the synthetic datasets. Table 2 shows that the NMI for the row and column clusters is very high, indicating that even at high levels of added noise, co-clustering with clusterwise models can recover the original clusters.

Dataset	NMI (row)	NMI (column)
Dataset 1	1	0.9154
Dataset 2	0.8445	0.619
Dataset 3	0.67	0.7124
Dataset 4	1	0.9426

Table 2: NMI with true labels

We now evaluate the ability of co-clustering with clusterwise classification models (Model CC) to classify unknown matrix values in the synthetic datasets. The data is split as 90% training and 10% test (missing) and the technique in section 4 is used to predict the class label of the missing values. The predicted labels are compared to the true labels and the classification quality is evaluated using precision, recall, F-measure and misclassification error. We compare this approach to the partitional co-clustering algorithm, Bregman co-clustering [BDG$^+$06], which uses only the matrix Z without any attribute information. The Bregman co-clustering algorithm is very flexible and can work with several distance measures and co-cluster definitions. The special case of Bregman co-clustering that we compare with uses squared Euclidean distance as the distance measure and tries to find uniform co-clusters that minimize the distance of the data points within the co-cluster to the co-cluster mean [3], since this case best matches the data generation process.

In order to apply co-clustering (CC) to this problem, in matrix Z, we encode positive class labels by the value 1 and negative by 0. The co-clustering algorithm approximates the cell values within each co-cluster by the co-cluster mean μ_{gh}. If a missing cell z_{ij} is assigned to row cluster g and column cluster h, with co-cluster mean μ_{gh}, we assign a class label to z_{ij} using the rule $z_{ij} = 1$ if $\mu_{gh} >$ threshold , $z_{ij} = -1$ otherwise. If the threshold is selected to be 0.5 this rule can be interpreted as assigning a missing cell the majority class label within its co-cluster. We also compare our approach with a single logistic regression classification model (Global Model), which is Model CC with $k = 1$ and $l = 1$.

Table 3 displays the precision, recall, F-measure and misclassification error for simultaneous co-clustering and classification (Model CC), co-clustering (CC) and a single logistic regression model (Global Model). The results are averaged over 5 random 90-10% splits of the data. The values in parentheses are the standard errors. The threshold is set to 0.5 for all these experiments since the same threshold was used to obtain class labels from probability values while generating the synthetic data. One can observe that on all the synthetic datasets Model CC does significantly better than CC and Global Model in terms of both the F-measure and the misclassification error.

On these datasets we also evaluate the ability of the simultaneous co-clustering and classification algorithm to reconstruct the original data matrix. We find that on all the datasets this approach is consistently able to recover a close approximation of the original data matrix. Additionally, the cluster assignments made by the algorithm closely match the true underlying

[3] This corresponds to scheme 2 of the Bregman co-clustering algorithm [BDG$^+$06] with squared Euclidean distance.

Algorithm	Log Loss	Precision	Recall	F-measure	Misclassification Error
Dataset 1					
Global Model	2025.969 (0)	0.894 (0.004)	0.952 (0.004)	0.922 (0.004)	0.131 (0.006)
CC	-	0.881 (0.007)	0.948 (0.003)	0.913 (0.003)	0.149 (0.006)
Model CC	764.270 (2.286)	**0.967 (0.003)**	**0.979 (0.002)**	**0.973 (0.002)**	**0.044 (0.002)**
Dataset 2					
Global Model	2410.106 (0)	0.883 (0.008)	0.927 (0.009)	0.904 (0.007)	0.151 (0.01)
CC	-	**0.910 (0.002)**	0.880 (0.004)	0.895 (0.003)	0.159 (0.004)
Model CC	1872.862 (13.981)	0.908 (0.006)	**0.937 (0.005)**	**0.922 (0.005)**	**0.124 (0.007)**
Dataset 3					
Global Model	30295.994 (0)	0.926 (0.001)	0.972 (0)	0.948 (0)	0.092 (0.001)
CC	-	0.935 (0.004)	0.956 (0.004)	0.945 (0.001)	0.096 (0.001)
Model CC	13923.973 (115.886)	**0.968 (0.002)**	**0.979 (0)**	**0.974 (0.001)**	**0.046 (0)**
Dataset 4					
Global Model	86777.68 (0)	0.928 (0)	0.971 (0)	0.949 (0)	0.09 (0)
CC	-	0.931 (0.003)	0.968 (0.002)	0.949 (0.001)	0.09 (0.001)
Model CC	29413.8086 (7.651)	**0.979 (0)**	**0.984 (0)**	**0.981 (0)**	**0.032 (0)**

Table 3: Comparison of classification performance.

cluster labels.

7.2 Recommender System Application

This application deals with data based on course choices made by MBA students, in the business school at UT Austin. The objective is to use the information of previous known course choices of students to predict unknown choices. These predictions can be used to recommend the right courses for students to take in the future. The data includes a matrix of students vs. courses with class labels = 1 if the student took the course and -1 otherwise. Each student has attributes including the student's career type (investment banker, corporate finance, IT, manager, consultant) and undergraduate degree. The course attributes include the department offering the course (accounting, finance, MIS, marketing, management), the course evaluation score and a binary variable indicating whether the course is quantitative. The dataset is skewed with unequal priors of the positive and negative classes. Around 25% of the student course choices are positive and the rest negative.

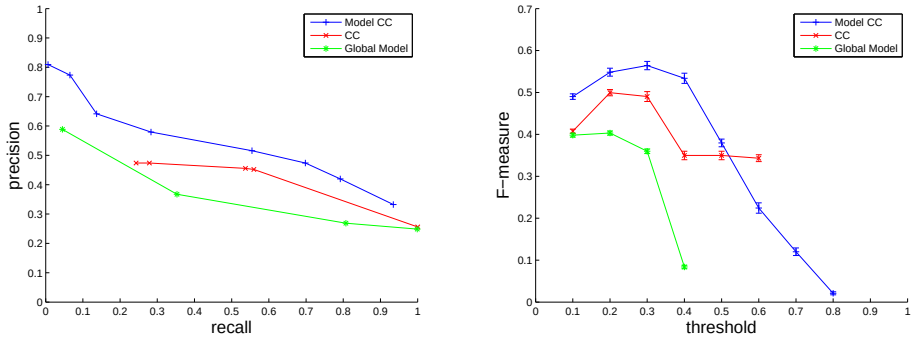
7.2.1 Classifying Missing Cell Values

In order to test the classification capability of Model CC on this problem, the data is split as 90% training and 10% test and the class labels in the test set, i.e. the unknown student-course choices, are predicted using the co-cluster models. Results are obtained by averaging over 10 random 90-10% data splits. We compare the Model CC results with CC and Global Model. For assigning class labels to the missing matrix entries we use a threshold that is varied from 0.1 to 0.9 to get a range of precision-recall tradeoffs.

Figures 3(a), 3(b), 3(c) display the precision-recall curves, the F-measure and the classification error of the 3 algorithms at different values of the threshold. Beyond a certain threshold, both CC and Global Model classify all the data points as belonging to the negative class, causing the F-measure to be undefined. Such points are excluded from the Precision-Recall curve

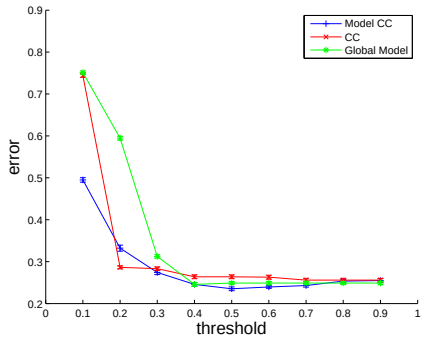
and the F-measure plot.

Model CC is significantly better than CC and Global Model in terms of precision and recall as can be seen in the Precision-Recall curves and hence its F-measure is consistently better than the other approaches at all values of the threshold. The classification error of Model CC is also lower than CC and Global Model. At threshold value 0.2 however, CC has a lower error as it has a much smaller number of false positives.



(a) Precision-Recall curve

(b) F-measure



(c) Classification error

Figure 3: Evaluation of classification of missing student-course choices in the recommender system application.

8 Experimental Evaluation of Regression Results

8.1 Synthetic Datasets

The algorithm described in section 5 is evaluated on a number of synthetic datasets. Each synthetic dataset is created by generating $k \times l$ blocks of data, each corresponding to a true co-cluster of real values generated by a linear regression model. Each block has a different linear regression model, with its co-efficient vector $\boldsymbol{\beta}$ set randomly. The blocks are then appended together in the form of a grid to form the data matrix Z. The cell value z_{ij} within each block is obtained by taking a linear combination of randomly generated customer and product

attributes with the co-efficient vector i.e. $z_{ij} = \boldsymbol{\beta}^T \mathbf{x_{ij}}$. We then add random gaussian noise to all the values in Z. Due to the added noise, the clusterwise linear models do not have a perfect fit. We quantify the linear relationships existing in the data in terms of the average R^2 of the linear models.

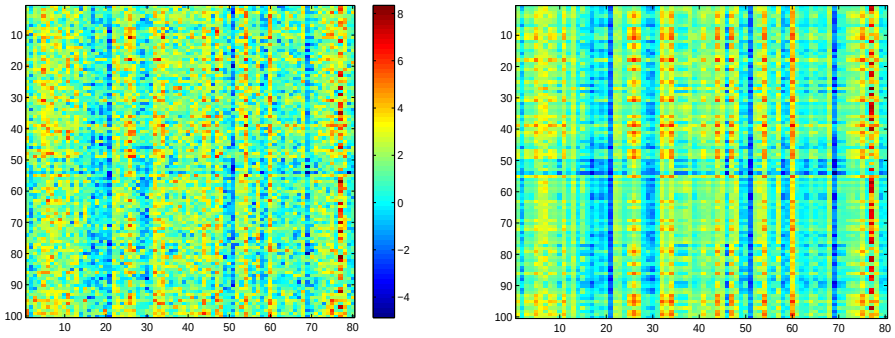
Table 4 describes the synthetic datasets [4] that were used for experimentation. Dataset 1 and 2 are very similar, dataset 2 has more noise and a weaker linear relationship as compared to dataset 1. Dataset 3 is larger, also with a substantial amount of noise. Dataset 5 is similar to dataset 4, but 0.1% of the data is perturbed to create outliers with large positive values.

	m,n	$\|C\|$,$\|P\|$	k,l	avg. R^2 of linear models
Dataset 1	100, 80	3,4	3,2	0.5670
Dataset 2	100, 80	3,4	3,2	0.4120
Dataset 3	500, 300	3,4	4,3	0.3964
Dataset 4	800, 500	5,5	8,6	0.3713
Dataset 5	800, 500	5,5	8,6	0.1732

Table 4: Synthetic Datasets

8.1.1 Results on Synthetic Data

Figure 4 displays the ability of the Model CC algorithm to reconstruct the original data matrix on synthetic dataset 1. The original data matrix is compared to the approximated matrix, obtained by the linear combination of row and column attributes using co-cluster model co-efficients. One can observe that the reconstruction is quite close to the original.



(a) Original Data Matrix (b) Reconstructed Data Matrix

Figure 4: Reconstruction ability of models

In order to evaluate the cluster assignments made by the Model CC algorithm, the NMI of the row and column cluster labels with the true labels is computed on the three synthetic datasets. Table 5 shows that the NMI for the row and column clusters is very high, indicating

[4] Datasets available at `http://www.ece.utexas.edu/~deodhar/modelCCData`

that even at high levels of added noise, co-clustering with clusterwise regression models can recover the original clusters.

Dataset	NMI (row)	NMI (column)
Dataset 1	1	1
Dataset 2	0.9583	0.9154
Dataset 3	0.6899	1
Dataset 4	0.9444	0.8587
Dataset 5	0.6187	0.6086

Table 5: NMI with true labels

We now evaluate the ability of co-clustering with clusterwise models to predict unknown matrix values. The data is split as 90% training and 10% test (missing) and the technique in section 5 is used to predict the missing values. The quality of the predictions is compared using mean squared error. Table 6 displays the prediction errors for a single linear regression model (Global Model), co-clustering, where no attribute information is used (CC), co-clustering with clusterwise models (Model CC) and the reduced parameter model (Reduced Model). The results are averaged over 5 random 90-10% splits of the data. Training Err. is the mean squared error on the training data and Test Err. is the mean squared error on the test data. The values in parentheses are the standard errors. Avg. R^2 is the average R^2 of the linear regression models constructed on the training data. One can observe that on the synthetic datasets Model CC does significantly better than the other approaches when the noise level is reasonable and there are no outliers (datasets 1-4), since the generative model of the data matches most closely with the data model assumed by Model CC. Under these conditions Reduced CC also does better than CC and Global Model but not as well as Model CC. However, in the presence of outliers (dataset 5) Model CC tends to overfit and Reduced CC does slightly better. On all the datasets, the average R^2 of the models reconstructed by Model CC is very close to that of the original models, whereas if we try to fit a single linear model its R^2 is substantially lower.

Figure 5 shows the plot of the predicted matrix values vs the actual values on synthetic dataset 1. For perfect prediction the plot will be a straight line at 45 degrees. The plot for simultaneous co-clustering and regression is closer to the ideal than that for co-clustering. In case of co-clustering the 6 co-cluster means are used to predict all the missing values and hence the predicted values are one of the 6 means. In simultaneous co-clustering and regression, however, even though the set of linear co-efficients is common for all the values in a co-cluster, each predicted value could be different due to different row and column attributes.

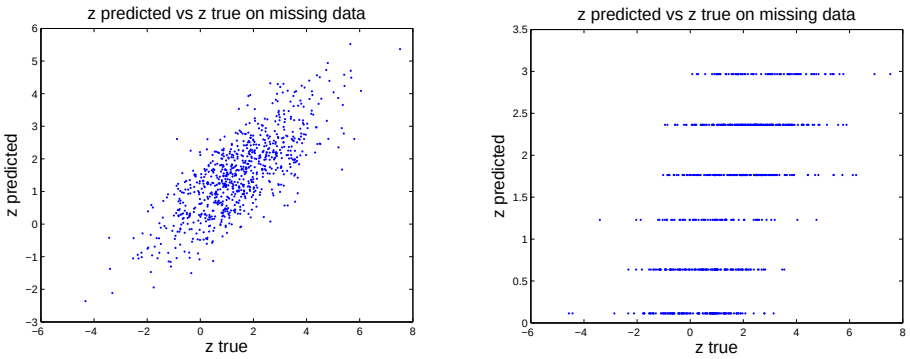
8.2 Real Marketing Dataset

We also applied the simultaneous co-clustering and regression approach to a challenging marketing application. Given purchase information for a set of customers and products along with customer and product attributes, we simultaneously clustered customers and products and used the co-clustering solution to predict unknown customer product purchase information. The obtained co-clusters also provide information about customer segments in the market and equivalent product groups, achieving simultaneous market segmentation and structure. The dataset that we use is the publicly available ERIM dataset [5] consisting of household panel

[5] URL:`http://www.gsb.uchicago.edu/kilts/research/db/erim/`

Algorithm	Training Err.	Test Err.	Avg. R^2
Dataset 1			
Global Model	1.399 (0.002)	1.369 (0.020)	0.458
CC	1.67 (0.003)	1.697 (0.027)	-
Reduced Model	1.276 (0.005)	1.291 (0.02)	0.258
Model CC	1.006 (0.002)	**1.024 (0.021)**	0.566
Dataset 2			
Global Model	2.803 (0.008)	2.843 (0.072)	0.332
CC	3.118 (0.006)	3.258 (0.078)	-
Reduced Model	2.647 (0.01)	2.770 (0.052)	0.171
Model CC	2.418 (0.028)	**2.596 (0.051)**	0.421
Dataset 3			
Global Model	2.23 (0.001)	2.223 (0.008)	0.286
CC	2.177 (0.001)	2.184 (0.009)	-
Reduced Model	2.0250 (0.002)	2.0291 (0.011)	0.18
Model CC	1.79 (0.017)	**1.789 (0.019)**	0.441
Dataset 4			
Global Model	3.719 (0.002)	3.709 (0.017)	0.274
CC	3.729 (0.004)	3.749 (0.021)	-
Reduced Model	3.638 (0.002)	3.651 (0.016)	0.147
Model CC	3.1102 (0.015)	**3.114 (0.025)**	0.368
Dataset 5			
Global Model	12.9798 (0.089)	13.085 (0.804)	0.115
CC	12.8990 (0.086)	13.318 (0.798)	-
Reduced Model	12.851 (0.089)	**13.033 (0.803)**	0.064
Model CC	12.363 (0.107)	13.064 (0.851)	0.258

Table 6: Prediction error on missing data.



(a) Simultaneous co-clustering and regression (Model CC)

(b) Co-clustering (CC)

Figure 5: Predicted values vs actual values.

data collected by A.C. Nielsen, which is well known in the marketing community and has been used by several researchers [KR94, KS98, SAC99]. This dataset has purchase information for six product categories over a period of 3 years from 1985-1988 for households in Sioux Falls, South Dakota. The dataset includes household demographics as well as product characteristics.

Detailed information about each shopping visit including the total expenditure and advertising information is also recorded.

The data preprocessing steps we took are similar to the data selection procedure by Seetharam et al. [SAC99]. We have 6 product categories (ketchup, tuna, sugar, tissue, margarine and peanut butter) with a total of 121 products. Brands with very low market share in each product category have been omitted. We select households that made at least 2 purchases in each product category, resulting in a set of 1714 households. We select 6 household attributes - income, number of residents, male head employed, female head employed, total visits and total expense and 3 product attributes - market share, price, number of times the product was advertised. The data can be represented by a data matrix of households and brands where the cell values are the number of units of a brand purchased by a household, aggregated over the time the household was tracked. The number of units purchased can be used as an indicator of brand preference.

8.2.1 Dataset Properties

The data matrix is extremely sparse, with 74.86% of the values being 0. The distribution of the number of units purchased is also very skewed as can be seen in the histogram of the matrix entries in Figure 6. 99.12% of the values are below 20, while the remaining values are very large and range upto around 200. These few, large values could be considered as outliers with respect to the rest of the data.

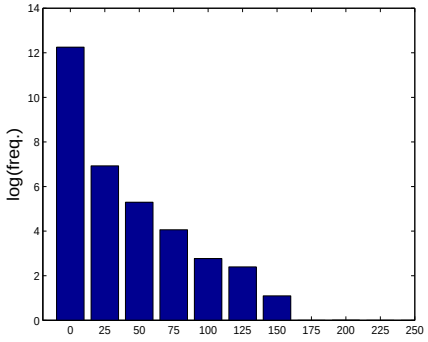


Figure 6: Histogram of Data Values

8.2.2 Standardization of the Data

The dimensions (items) in this application are products from 6 different product categories. The product attributes such as price and extent of advertising could vary from one category to another. When we construct a linear model for a co-cluster we weigh the attributes of all the products in the co-cluster by the same set of co-efficients. However, the products in the co-cluster could be from different categories with very different ranges of attribute values. We hence need to standardize the product attributes to make them comparable across categories. We transform each product attribute value a to $a^{'} = \frac{a-\mu_c}{\sigma_c}$, where μ_c and σ_c are the mean and standard deviation within the corresponding product category c. This problem does not arise

in case of the customer attributes since they are relatively comparable. The matrix cell values, which are the number of units purchased could also be very different across categories and have to be standardized. The cell values z_{ij} within each sub-matrix of all the products belonging to a specific product category c are transformed to $z'_{ij} = \frac{z_{ij} - \mu_{z_c}}{\sigma_{z_c}}$ where μ_{z_c} and σ_{z_c} are the mean and standard deviation of the all the values in the sub-matrix. Since the standardization of the data is a linear transformation, the dataset properties described in section 8.2.1 continue to hold.

8.2.3 Data Reconstruction

Table 7 displays the mean squared error of the approximated data matrix obtained by the different algorithms on the entire standardized dataset, averaged over 10 runs. "Row Clustering" is Model CC with the number of column clusters set to 1 and "Column Clustering" is Model CC with the number of row clusters set to 1. Note that Model CC obtains the best reconstruction of the original matrix as compared to the other approaches in terms of MSE (mean squared error). Table 7 also shows the average R^2 of the linear models constructed within each co-cluster. The R^2 values are actually quite low, indicating that a strong linear relationship does not really exist in the data, which is to the disadvantage of the simultaneous co-clustering and regression algorithm.

Algorithm	MSE	Avg. R^2
Global Model (k=1,l=1)	0.930 (0)	0.0696
CC (k=10, l=4)	0.842 (0.003)	-
Row Clustering (k=10, l=1)	0.887 (0)	0.0896
Column Clustering (k=1, l=4)	0.88 (0.001)	0.0714
Reduced Model (k=10, l=4)	0.876 (0)	0.0426
Model CC (k=10, l=4)	**0.794 (0.005)**	0.1308

Table 7: Reconstruction error on entire ERIM dataset.

8.2.4 Predicting Unknown Data Values

The prediction error of the different approaches on this problem is computed by averaging over 10 random 90-10% training and test data splits. Table 8 shows the training error (Training Err.), the test set error (Test Err.) on the standardized data and the test set error on the original, unstandardized dataset obtained by back transforming the standardized data (Test Err. Original). Mean R^2 is the mean R^2 of the regression models.

As mentioned in section 8.2.1 the data is skewed with a few very large outliers. In the presence of outliers the clusterwise models overfit the training data and do not generalize well to the test data. Model CC is the most susceptible to overfitting since it is the most complex and involves the most number of parameters. Model CC does better than Global Model but slightly worse than CC. Reduced Model does better than Model CC in this scenario since it has fewer parameters and a simpler overall model, which generalizes better. Reduced Model does slightly better than CC as well, illustrating that using the attribute information in the prediction process helps. However, this improvement is small, which is because the data does not show a very strong linear relation, indicated by the low R^2 values of the linear models.

Algorithm	Training Err.	Test Err.	Test Err. Original	Avg. R^2
Global Model (k=1,l=1)	0.931 (0.003)	0.928 (0.023)	16.776 (0.584)	0.067
CC (k=4,l=4)	0.853 (0.003)	0.905 (0.022)	15.501 (0.511)	-
Reduced Model (k=4,l=4)	0.878 (0.002)	**0.887 (0.023)**	**15.339 (0.553)**	0.056
Model CC (k=4,l=4)	0.823 (0.002)	0.905 (0.021)	15.688 (0.519)	0.113

Table 8: Comparison of prediction error on ERIM dataset.

Algorithm	Training Err.	Test Err.	Test Err. Original	Avg. R^2
Global Model (k=1,l=1)	0.913 (0.001)	0.920 (0.01)	4.24 (0.06)	0.086
CC (k=4,l=4)	0.833 (0.001)	0.890 (0.009)	4.002 (0.056)	-
Reduced Model (k=4,l=4)	0.849 (0.001)	**0.872 (0.009)**	**3.893 (0.052)**	0.074
Model CC (k=4,l=4)	0.804 (0.001)	0.883 (0.007)	3.965 (0.044)	0.138

Table 9: Prediction error on ERIM dataset (model for low valued entries).

Model CC has the highest R^2 and its training error is the least, however its test error is quite large, confirming the overfitting tendency of the models.

Linear "least squares" regression is very sensitive to outliers and a few outliers could influence the model very heavily and skew the results. Hence on this dataset, for a better comparison of linear model based techniques with respect to prediction of missing values, we need some way of dealing with outliers. The data matrix includes two very different sets of values for the number of units purchased, low values (below 20 units purchased) that form the bulk of the data and a few high values. It is unreasonable for a model based on linear regression to capture both small as well as extremely large values (outliers) simultaneously and a more suitable approach would be to separate out these two very different sets of values and model them independently. A threshold of 20 for the number of units purchased was used to separate the bulk of the matrix entries (99.12%) from the tail of high values. This separation can easily be handled by the co-clustering algorithm by setting the weight of the outlier entries in the data matrix to 0. The standardization steps described in Section 8.2.2 are carried out on the separated lower valued matrix entries. We first focus on the model and the prediction problem associated with these entries.

Model for low valued matrix entries: The different modeling approaches on this problem are evaluated by comparing their average prediction accuracy over 10 random 90-10% training and test data splits. Table 9 shows the training mean squared error and the test set error for the different approaches. Model CC and Reduced Model do significantly better than Global Model on the test set. Reduced Model also does better than CC.

Separate Models for low and high valued entries: Our complete model for the dataset consists of a model for the low valued matrix entries as described above and a separate model for the high valued outliers. The training data is divided into low and high valued entries using the threshold of 20 number of units purchased and a different model is trained on each set of values. For modeling the low values we compare the performance of co-clustering based models (Model CC, CC) with $k = 4$ and $l = 4$ and a single regression model (Global Model). The model for the high values is a linear regression model. The reason for choosing a very simple model for the high values is that these values are very few in number and have a skewed distribution, which may cause more sophisticated models to overfit. The attributes of the training data points are used as features to train a classifier, which when given a new set of attributes will

be able to classify the corresponding cell value as high or low. The appropriate model can then be used to predict the unknown value. This classification problem is also difficult since the two classes (high and low) are heavily imbalanced, with the low class comprising more than 99% of the training data. Most classifiers like decision trees and logistic regression get good overall accuracy by simply predicting the target class for all the test points as the low class. Different costs could be assigned for misclassifying the two classes since we would like all the high points to be classified correctly, but would be tolerant to a few low points being misclassified as high. We found k nearest neighbors with $k = 3$ to work reasonably in practice, although slow, with an average accuracy of 0.9909. Figure 7 depicts the procedure outlined above.

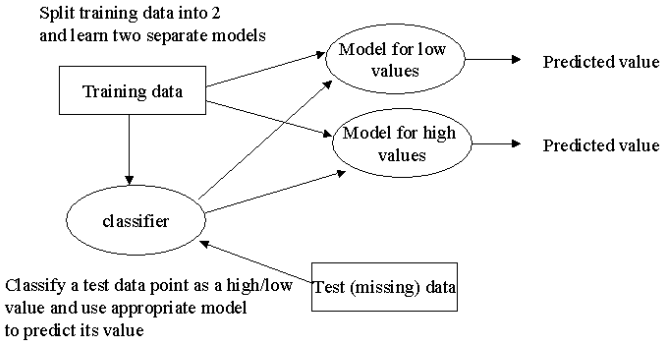


Figure 7: Modeling the data entries with 2 separate models

Table 10 compares the performance of the above approach using Model CC, CC and a single model for the modeling the low values. Training Err. Low and Training Err. High are the training errors for the low and high value models and Test Err. and Test Err. Original represent the overall test set error over all the missing values. The approaches that use Model CC and Reduced Model do significantly better than the Global Model and CC approaches. Overall the prediction error is higher than in 8, which is mainly due to the classification errors that cause low values to be predicted by the high valued model.

Algorithm	Training Err. Low	Training Err. High	Test Err.	Test Err. Original
Global Model	0.459 (0.001)	16.931 (0.223)	1.001 (0.027)	19.332 (0.579)
CC	0.417 (0.001)	20.011 (0.214)	0.987 (0.027)	19.777 (0.561)
Reduced Model	0.427 (0.001)	16.931 (0.223)	**0.970 (0.027)**	**18.647 (0.553)**
Model CC	0.399 (0.001)	16.931 (0.223)	0.972 (0.027)	18.688 (0.577)

Table 10: Prediction error on ERIM dataset (separate models for high and low valued entries).

We now present an alternative way of dealing with outliers, which gives outliers very low weight while constructing the models.

Weighting outliers less: An alternative way of dealing with outliers is to reduce the influence of the extremely high values on the constructed models, allowing them to generalize better. This can be done by giving high valued matrix entries a very small fixed weight, enabling the linear models to focus on the bulk of the data, rather than fit a few high values. In this approach we consider all the matrix entries together, but set the weight for entries greater than 5 (after standardization) to a small value selected using 10 fold cross validation. The experimental results are displayed in Table 11. Model CC now uses weighted least squares,

which reduces the overfitting problem, and Model CC and Reduced Model do better than a single model. Model CC now does better than Reduced Model as well.

Algorithm	Training Err.	Test Err.	Test Err. Original	Avg. R^2
Global Model (k=1,l=1)	0.385 (0)	0.919 (0.027)	17.141 (0.535)	0.091
CC (k=4,l=4)	0.337 (0)	0.886 (0.027)	15.873 (0.518)	-
Reduced Model (k=4,l=4)	0.345 (0)	0.880 (0.027)	15.823 (0.501)	0.084
Model CC (k=4,l=4)	0.324 (0.001)	**0.8769 (0.028)**	**15.771 (0.535)**	0.148

Table 11: Prediction error after weighting outliers less.

Summary of Empirical Studies: Based on the results in Sections 7 and 8 we observe that the simultaneous co-clustering and prediction approach holds promise for both classification and regression problems, at least for the datasets examined so far. One should be able to further improve the results by using alternative prediction models within each co-cluster that more closely conform to the data characteristics. This is particularly true for the ERIM dataset which is known to have significant outliers, but we still used linear least squares regression as it is widely adopted and understood. Note that overfitting of the models to outliers is addressed to some extent by the reduced parameter model. Further improvements can be achieved by using a more robust error function rather than squared error [Hub81]. Moreover, non-linear models would help because the limitations of linear models for this dataset is quite evident by the low R^2 values obtained by several researchers who previously applied such models to ERIM.

9 Concluding Remarks

Simultaneous co-clustering and modeling is a promising framework that generalizes co-clustering, collaborative filtering and traditional segment-wise modeling. Note that this approach is not limited to the algorithms presented in Sections 4 and 5, but forms a broad framework for solving difficult classification and regression problems in general. The logistic regression models, for instance, could easily be replaced by other classifiers by modifying the objective function to optimize the corresponding loss function.

While this paper concentrated on synthetic and marketing data for illustration, there are many other domains characterized by data matrices supplemented by annotated row and column entities. For example, this approach can be used to analyze microarray data with gene and experiment annotations, social network settings with sets of attributes attached to persons (rows) and relationships (columns), and clustering of web documents annotated by link as well as semantic information. It will be worthwhile to investigate specific instances of this framework (with specific choices of the co-clustering model used as well as of the classifier/predictor used) in terms of their suitability for different problems within this wide range of application domains.

Acknowledgments: The research was supported by NSF grants IIS 0325116 and IIS 0307792. We would like to thank Prof. McAlister and Andrea Godfrey from the UT Business school for the MBA dataset and insightful discussions.

References

[BDG+06] A. Banerjee, I. Dhillon, J. Ghosh, S. Merugu, and D. Modha. A generalized maximum entropy to bregman co-clustering and matrix approximation. *To appear in JMLR*, pages Downloadable from `http://pegasus.ece.utexas.edu/~ghosh/jmlr--coclust.pdf`, 2006.

[BG93] T. Baumann and A. Germond. Application of the kohonen network to short-term load forecasting. In *ANNPS*, pages 407–412, 1993.

[CC00] Y. Cheng and G. M. Church. Biclustering of expression data. In *ICMB*, pages 93–103, 2000.

[CDGS04] H. Cho, I. S. Dhillon, Y. Guan, and S. Sra. Minimum sum squared residue co-clustering of gene expression data. In *SDM*, 2004.

[CMHD03] Lucia Conde, Alvaro Mateos, Javier Herrero, and Joaquin Dopazo. Improved class prediction in dna microarray gene expression data by unsupervised reduction of the dimensionality followed by supervised learning with a perceptron. *Journal of VLSI Signal Process. Syst.*, 35:245–253, 2003.

[DBSP93] M. Djukanovic, B. Babic, D. Sobajic, and Y. Pao. Unsupervised/supervised learning concept for 24-house load forecasting. *IEE Proceedings-Generation, Transmission and Distribution*, 140:311–318, 1993.

[DMM03] I. Dhillon, S. Mallela, and D. Modha. Information-theoretic co-clustering. In *KDD*, pages 89–98, 2003.

[GM05] T. George and S. Merugu. A scalable collaborative filtering framework based on co-clustering. In *ICDM*, pages 625 – 628, 2005.

[HKBR99] J. Herlocker, J. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *SIGIR*, pages 230–237, 1999.

[Hub81] P. J. Huber. *Robust Statistics*. Wiley, New York, 1981.

[JJ94] M.I. Jordan and R.A. Jacobs. Hierarchical mixture of experts and the EM algorithm. *Neural Computation*, 6:181–214, 1994.

[JY03] Rebecka Jornsten and Bin Yu. Simultaneous gene clustering and subset selection for sample classification via mdl. *BMC Bioinformatics*, 19(9):1100–1109, 2003.

[KR94] B. Kim and P. Rossi. Purchase frequency, sample selection, and price sensitivity: The heavy-user bias. *Marketing Letters*, pages 57 – 67, 1994.

[KS98] B. Kim and M. Sullivan. The effect of parent brand experience on line extension trial and repeat purchase. *Marketing Letters*, pages 181 – 193, 1998.

[LKM05] Xiaoxing Liu, Arun Krishnan, and Adrian Mondry. An entropy-based gene selection method for cancer classification using microarray data. *BMC Bioinformatics*, 6:76, 2005.

[LL03] W. Lee and B. Liu. Learning with positive and unlabeled examples using weighted logistic regression. In *ICML*, 2003.

[LS00] Larisa Lokmic and Kate A. Smith. Cash flow forecasting using supervised and unsupervised neural networks. *IJCNN*, 06:6343, 2000.

[MO04] S. C. Madeira and A. L. Oliveira. Biclustering algorithms for biological data analysis: A survey. *IEEE Trans. Computational Biology and Bioinformatics*, 1(1):24–45, 2004.

[OH01] K. Oh and I. Han. An intelligent clustering forecasting system based on change-point detection and artificial neural networks: Application to financial economics. In *HICSS-34*, volume 3, page 3011, 2001.

[SAC99] P.B. Seetharaman, A. Ainslie, and P.K. Chintagunta. Investigating household state dependence effects across categories. *Journal of Marketing Research*, pages 488 – 500, 1999.

[SG02] A. Strehl and J. Ghosh. Cluster ensembles – a knowledge reuse framework for combining partitionings. *JMLR*, 3(3):583–617, 2002.

[Sha96] A. Sharkey. On combining artificial neural networks. *Connection Science*, 8(3/4):299–314, 1996.

[SS04] A. Sfetsos and C. Siriopoulos. Time series forecasting with a hybrid clustering scheme and pattern recognition. *Systems, Man and Cybernetics, Part A, IEEE*, 34:399– 405, 2004.

[WS91] M. Wedel and J. Steenkamp. A clusterwise regression method for simultaneous fuzzy market structuring and benefit segmentation. *Journal of Marketing Research*, pages 385 – 396, 1991.

[YFW06] Y. Ying, F. Feinberg, and M. Wedel. Leveraging missing ratings to improve online recommendation systems. *Journal of Marketing Research*, pages 355 – 365, 2006.

[ZNB05] Shuai Zhang, Daniel Neagu, and Catalin Balescu. Refinement of clustering solutions using a multi-label voting algorithm for neuro-fuzzy ensembles. In *ICNC*, pages 1300–1303, 2005.