Improving Response Prediction for Dyadic Data

Stat 598T "Statistical Network Analysis" Project Report

Nik Tuzov*

April 2008

Keywords: Dyadic data, Co-clustering, PDLF model, Neural networks, Unsupervised learning, Recommendation systems, Internet advertising

* Purdue University, Department of Statistics, Computational Finance Program
Email: ntuzov@purdue.edu
Web: http://www.stat.purdue.edu/~ntuzov/

1. Introduction

The most widely known predictive statistical models are those based on supervised learning approach when the response variable is modeled as a certain function of observed covariates. Popular examples include Multiple Linear Regression, Logistic Regression, Poisson Regression and other members of GLM family. Another example is the class of neural network models that typically have a greater predictive power but lower interpretability than GLM.

On the other hand, the unsupervised learning approach does not utilize any observed covariates to predict the response. In particular, a number of such methods have been proposed for so-called *dyadic data*. Each dyadic observation involves two objects taken from two different sets and the observed response associated with each pair. For instance, a pair of user (indexed by i), movie (indexed by j) and the rating (denoted y_{ij}) that user i gives to the movie j are an example of a dyad and the corresponding response. All possible pairs are represented by the dyadic matrix with the number of rows equal to the number of users and the number of columns equal to the number of different movies. The values in the matrix correspond to the values of y_{ij}. According to Agarwal and Merugu [1], such data structures are encountered in social networks, recommendation systems, Internet advertising, and so on.

Typically, the majority of y_{ij} are just missing values (e.g., in the example above, an average user rates only a small fraction of all movies). An unsupervised learning method is essentially trying to fill in the missing observations based on the non-missing information derived from all (i, j, y_{ij}). The examples of such methods are: Co-clustering with Bregman divergences [2], Cross-association learning [3], Singular value decomposition [6], Non-negative matrix factorization [7] and Information-theoretic co-clustering [5].

A supervised learning method, on the other hand, fills in the missing data based on (x_{ij}, y_{ij}) where x_{ij} is the observed covariate (such as movie type and user demographic information). While the former method disregards the possible contribution of covariates, the latter ignores the local structure of the dyadic space (Agarwal and Merugu, [1]).

Therefore, when dealing with dyadic data it is desirable to combine both approaches in order to enhance the predictive accuracy. A good solution has been proposed in Agarwal and Merugu [1] and called PDLF – "Predictive Discrete Latent Factor" model. PDLF includes a supervised part (represented by an arbitrary GLM model) and an unsupervised part (represented by a method closely related to Co-clustering with Bregman divergences).

The proposed approach is scalable, computationally efficient, interpretable (w.r.t. GLM part), and, importantly, allows for the discovery of new useful covariates that can be derived from the unsupervised part.

Agarwal and Merugu illustrate their method by constructing movie recommendation systems that can predict both binary and continuous movie ratings based on known values of (i, j, x_{ij}, y_{ij}) which are obtained from the MovieLens dataset (see Part 3).

This project is aimed at extending the PDLF model by modifying the supervised learning-based part. In particular, for a movie recommendation system, the prediction of movie rating is the main objective. Therefore, using a neural network model instead of GLM can pay off despite the low (or even non-existing) interpretability of the resulting model. The next section describes the proposed method and shows how it is related to PDLF.

2. Methodology

The PDLF model is based on the following framework:

$Y = [y_{ij}]$ - response matrix, $i = 1, m \quad j = 1, n$

x_{ij} - s-dimensional observed covariate

K*L - number of clusters or "blocks" in the response matrix

δ_{IJ} – scalar "offset" or "interaction term" for cluster (I, J), I = 1,K J = 1,L

π_{IJ} - proportion of $[y_{ij}]$ belonging to cluster (I, J)

$f(y_{ij}; \theta)$ - probability density of a GLM model; θ is a scalar parameter

β - s-dimensional vector of regression coefficient of GLM model

Then the conditional density of y_{ij} is represented as:

$$p(y_{ij} \mid x_{ij}) = \sum_{I,J} \pi_{IJ} \cdot f(y_{ij}; \beta^t x_{ij} + \delta_{\text{IJ}})$$

Therefore, the prediction is based on both the observed covariates ($\beta^t x_{ij}$ term) and latent factors represented by the clustering parameters δ_{IJ}, π_{IJ}. While parameter θ for the "pure" GLM model is $\beta^t x_{ij}$, the PDLF model adjusts that value by δ_{IJ} when y_{ij} falls into the cluster (I, J). It may be assumed that y_{ij} can belong to many clusters with certain probabilities summing up to one ("soft assignment") or that y_{ij} can belong to one and only one cluster ("hard assignment"). The "hard" version is used to analyze real datasets (because of scalability restrictions) so only the "hard" version will be considered below.

The co-clustering approach assumes that clusters in the dyadic matrix are rectangular blocks based on "row" clusters and "column" clusters, i.e.

y_{ij} belongs to cluster $(\rho(i), \gamma(j))$, where

$\rho(i)$ = I, I takes values from 1 to K

$\gamma(j)$ = J, J takes values from 1 to L

In order to fit the model, Agarwal and Merugu propose a generalized EM algorithm that includes fours steps. In the case of "hard" assignment, the steps reduce to maximizing the total log-likelihood with respect to:

1) $\{\delta_{IJ}\}$, K*L - dimensional
2) β, s-dimensional
3) $\rho(.)$, m-dimensional
4) $\gamma(.)$, n-dimensional

At each step, the likelihood is maximized w.r.t. the corresponding parameter holding the other three parameters fixed.

In this project, a neural network is used instead of GLM. While GLM is estimated via maximum likelihood, the network parameters β (also called "weights") are estimated by minimizing the mean squared error (MSE) or, equivalently, the total sum of squared errors. Correspondingly, the network-based version of "hard" PDLF algorithm takes the following shape:

Denote: y_{ij} – observed (non-missing) reponse

$(\hat{y}_{ij} \mid x_{ij}, \beta)$ - fitted response from the neural network, given covariate and estimated weights β

The predicted response for dyad (i, j) is: $(\hat{y}_{ij} \mid x_{ij}, \beta) + \delta_{\rho(i),\gamma(j)}$

Input: Response matrix $Y = [y_{ij}]$, i = 1,m j = 1,n

Covariates $X = [x_{ij}]$, x_{ij} is s-dimensional

Number of row clusters: K

Number of column clusters: L

Output: Neural network weights β (dimension depends on the network),

K*L offsets $\{\delta_{IJ}\}$

$\rho(.)$, $\gamma(.)$ – row and column cluster assignments, m and n-dimensional, that minimize overall sum of squared differences between the observed and predicted response for dyad (i, j).

Method:

Initialize: $\rho(.)$, $\gamma(.)$ - randomly

$\delta_{IJ} = 0$

β - by fitting the network based just on Y and X

Repeat

Step 1: Keeping β's fixed, update offsets (interaction effects) δ_{IJ} :

For each I = 1,K , J = 1, L

$$\delta_{IJ} \xleftarrow[\text{argmin } \delta]{} \sum_{i \in I,\ j \in J} (y_{ij} - (\hat{y}_{ij} \mid x_{ij}, \beta) - \delta)^2$$

that is,

$$\delta_{IJ} = mean_{i \in I,\ j \in J}(y_{ij} - (\hat{y}_{ij} \mid x_{ij}))$$

Step 2: Keeping δ's fixed, update neural network weights, β :

$$\beta \xleftarrow[\text{argmin } \beta]{} \sum_{\text{all i, j}} (y_{ij} - (\hat{y}_{ij} \mid x_{ij}, \beta) - \delta_{\rho(i),\gamma(j)})^2,$$

which amounts to fitting the network with

covariates x_{ij} and "adjusted" response values $(y_{ij} - \delta_{\rho(i),\gamma(j)})$

Step 3: Update row cluster assignments: for each i = 1, m

$$\rho(\text{i}) \xleftarrow[\text{argmin I}]{} \sum_{\text{j = 1,n}} (y_{ij} - (\hat{y}_{ij} \mid x_{ij}, \beta) - \delta_{I\gamma(j)})^2$$

Step 4: Update column cluster assignments: for each j = 1, n

$$\gamma(\text{j}) \xleftarrow[\text{argmin J}]{} \sum_{\text{i = 1,m}} (y_{ij} - (\hat{y}_{ij} \mid x_{ij}, \beta) - \delta_{\rho(i)J})^2$$

Until convergence

Return $(\beta,\ \{\delta_{IJ}\},\ \rho,\ \gamma)$

The predicted response for dyad (i, j) is: $(\hat{y}_{ij} \mid x_{ij}, \beta) + \delta_{\rho(i),\gamma(j)}$

Simply stated, this algorithm alternates between the unsupervised co-clustering on the residuals obtained from the neural network, $y_{ij} - (\hat{y}_{ij} \mid x_{ij})$, and fitting the (supervised) network after the co-clustering effects are taken into account.

Note also that it is impossible to update $\rho(i)$ and $\gamma(j)$ unless there is at least one non-missing observation in ith row/ jth column, correspondingly.

Although the squared error loss function is more appropriate for continuous response, it can also be used for categorical response. In particular, there is practically no difference in ROC area between the logistic model fitted by MLE and its neural network version fitted with MSE (see Section 4).

3. Data description

MovieLens dataset [9] consists of 100000 ratings (on 1-5 scale) submitted by 943 users on 1682 movies between 09/1997 and 04/1998. Time of rating submission is recorded. Also, it contains the following covariate information:

User:

1) Age in years (continuous)
2) Gender (binary)
3) Occupation (categorical with 21 levels)
4) Zip code.

Movie:

1) Title
2) Release date
3) URL
4) Genres (19 levels; movie can belong to more than one genre)

Agarwal and Merugu do not mention how they transformed the covariates, but in this project the following is done:

1) User age is kept as it is
2) Movie age (in years) is computed as the difference between the release date and the rating date

All categorical covariates are coded on 0-1 basis; one level is dropped to avoid linear dependence among covariates. In particular:

3) Gender is a single binary covariate
4) Genres correspond to 18 binary covariates
5) Zip code. The first digit (0-9) of zip corresponds to one of 10 locations in USA + non-US (mostly Canadian) codes = 11 levels = 10 binary covariates
6) Occupation corresponds to 20 binary covariates

Altogether, there are 51 covariates plus the intercept. The original ratings were converted to 0/1 : 1 where the original rating > 3 and 0, otherwise.

In this large dataset, the distribution of the number of ratings per movie is:

Mean	59.51786
Median	27.00000
Std Deviation	80.40960

Quantile	Estimate
100% Max	583
99%	378
95%	230
90%	169
75% Q3	80
50% Median	27
25% Q1	6
10%	2
5%	1
1%	1
0% Min	1

The distribution of the number of ratings per user is:

Mean	105.9692
Median	64.5000
Std Deviation	100.95437

Quantile	Estimate
100% Max	737.0
99%	448.0
95%	311.0
90%	245.0
75% Q3	148.0
50% Median	64.5
25% Q1	33.0
10%	23.0
5%	21.0
1%	20.0
0% Min	20.0

Agarwal and Merugu used only 20000 ratings out of 100000 available, but they do not explain how the selection was done. It seems natural to select the most active users and the most rated movies (so that the dyadic matrix is not that sparse) but also to keep the sample representative of the average user (e.g. users with over 300 ratings are atypical).

Given that, a subset of 20603 ratings with 346 users and 966 movies was selected with 1 to 198 ratings per movie and 32 to 105 ratings per user. This appears to be quite representative

of the original dataset. Also, the sample size and number of users/movies are comparable to those in [1].

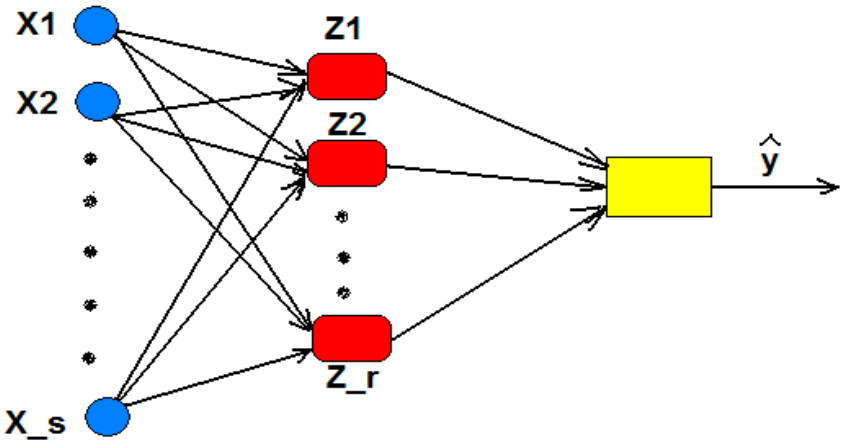
The next step is to split the 20603 observations into the training and validation datasets. Since the number of observations is large, k-fold cross-validation does not seem necessary. Note also that in order to get a training dataset, one cannot just delete any arbitrary observation because the training dataset still has to have at least one non-missing observation in each row and column. Moreover, since the training dataset has to be representative of the population, it is not desirable to delete too many observations from each particular row/column.

Therefore, 5700 observations (about 28%) were selected out of 20603 at random but under the condition that not more that 30% of observations in each row/column are selected. This validation sample corresponds to 346 users and 772 movies. The remaining training sample has 14403 observations, with 23 to 78 ratings per user and 1 to 139 ratings per movie.

The training dataset was checked for linear dependency among covariates and as a result one covariate (corresponding to occupation = “none”) was removed which left 50 covariates plus the intercept.

4. Selection of neural network model.

Since our problem is that of binary classification, a standard one-hidden layer, r-hidden nodes feed-forward neural network with sigmoid transfer function is utilized. The network structure can be depicted as follows:



where

$$Z_l = \sigma(\alpha_{0l} + \alpha_l^t \cdot \vec{X}),\ \ l = 1, r \text{ - "derived features"}$$

$$\hat{y} = \sigma(\theta_0 + \theta_1^t \cdot \vec{Z}) \text{ - fitted probability of } \{y = 1\}$$

$$\sigma(v) = 1/(1 + e^{-v}) \text{ - sigmoid function}$$

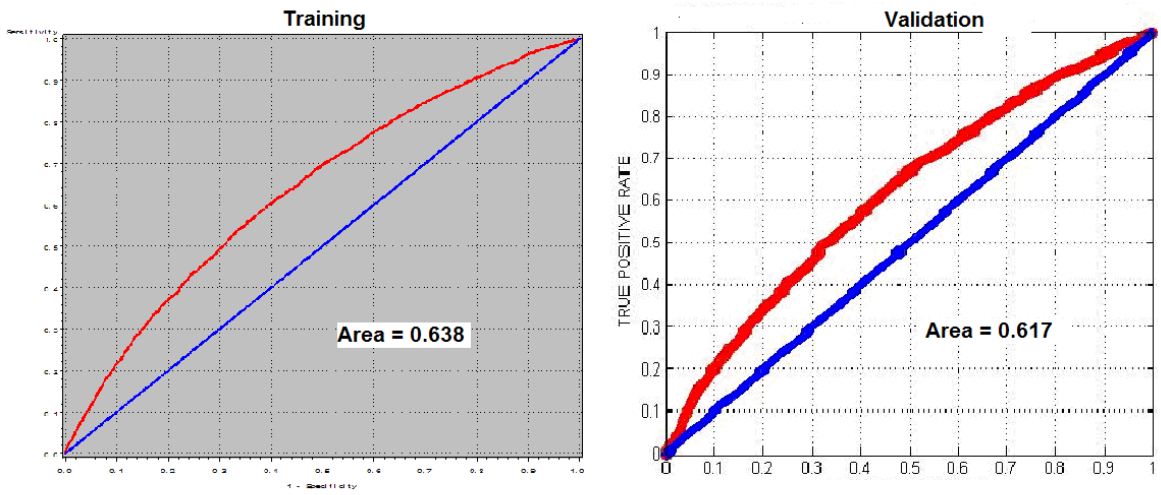
$$\text{Total number of parameters ("weights"): } (s + 1)r + (r + 1)$$

Note that when r = 1 the outer yellow layer is redundant and the network reduces to standard logistic regression which makes it possible to compare the results of this model with that of [1]. The only small difference is that the network uses the MSE instead of the likelihood function for training.

It is known that such network can approximate any continuous function mapping the X space into (0, 1) interval, given that r is large enough. In general, it is possible to obtain a more efficient model by introducing extra hidden layers and partial connectivity but such design requires some specific domain knowledge (e.g., in image recognition), which is not available for this particular application.

While MSE is technically convenient for the purpose of training, the area under Receiver Operating Characteristic (ROC) curve a lot more interpretable in this setting. It is going to be used below for model comparison (over the validation dataset).

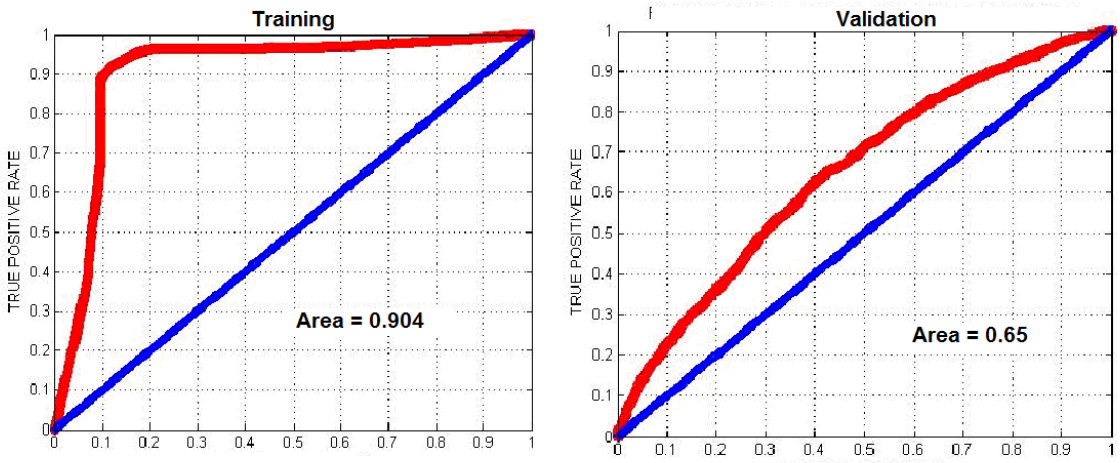
Below are shown two ROC curve plots that correspond to standard logistic regression fitted with MLE (no co-clustering so far).



Logistic regression ROC curves

The corresponding MSE-based neural network produces very similar results with the validation ROC area of about 0.62. Apparently, there is no good fit, which is also confirmed by the logistic deviance statistic (not shown).

On the other hand, a network with 70 hidden nodes (w/o co-clustering) has a great training performance and, although clearly overfitted, tells us that 70 nodes in the hidden layer are more than enough:



ROC curves for network with 70 hidden nodes

It turns out that even 40 hidden nodes are enough to produce an overfitted model (training ROC = 0.887, validation ROC = 0.663) within a reasonable training time. On the other hand, a smaller number of nodes results in slow convergence of MSE and therefore we'll stick to a 40-node network. To avoid overfitting in Step2 of the algorithm, the "early stopping" rule is used: weights are initialized around zero and move away from zero as the network is trained. As soon as the validation error starts to rise, the process is stopped. That way the effective number of parameters turns out much smaller than $(s + 1)r + (r + 1)$. Besides, the overall model validation ROC is monitored to avoid overfitting the entire model.

The network is trained with a gradient descent-like method, and for feed-forward networks *Matlab R2007a* has about 14 "training functions" corresponding to different versions of the gradient descent. For this particular network, *trainrp* turns out to be the most suitable since it takes into account the nature of sigmoid function used in this problem. Also, the fitted weights are dependent on the initial point of the gradient descent. It is customary to try a number of random initial weights and then average the predicted response over all of the trained networks. Averaging over 2 to 30 networks has been tried for this dataset and it appears that 10 networks are enough.

5. PDLF – Neural Network Model : Results

A number of different models have been fitted. The results are presented below:

	Logistic regression	Neural network	PDLF-Logistic	PDLF-Neural	PDLF-Neural	PDLF-Neural
Clusters	N/A	N/A	4 * 4	4 * 4	6 * 6	3 * 4
Hidden nodes	1	40	1	40	40	40
Validation ROC	0.62	0.6742	**0.6913**	**0.7128**	0.6919	0.708
Max. cluster size	N/A	N/A	2022	1913	5184	1847
Min cluster size	N/A	N/A	274	412	5	709
Max delta	N/A	N/A	0.25	0.13	0.23	0.02
Min delta	N/A	N/A	-0.4	**-0.57**	-0.36	-0.62

The best model appears to have 4 row and 4 column clusters (16 blocks) and 40 hidden nodes in the neural part (ROC = 0.7128). The first question is how it is compared to PDLF-Logistic model of Agarwal and Merugu. Since logistic regression is equivalent to having just one hidden node, the comparison can be easily done. It turns out that the validation ROC of PDLF-Logistic model (0.6913) is only slightly lower than that of the best model.

Also note that the values of δ_{IJ} in the table have a large magnitude that implies that in some clusters of the dyadic matrix the neural network-based model has a poor fit. In particular, the negative offset of -0.57 from the best model corresponds to a cluster of 756 observations produced by some 78 users and 253 movies where the neural network grossly overestimates the probability of {y = 1} (in the training dataset). In fact, the average observed rating is 0.117 whereas the average $(\hat{y}_{ij} \mid x_{ij}, \beta)$ rating is 0.696, the difference being the offset of about -0.57.

Agarwal and Merugu proposed to identify new covariates by looking into the features of such outstanding clusters. The 253 movies in the cluster range in release date from 1922 to 1998, for instance:

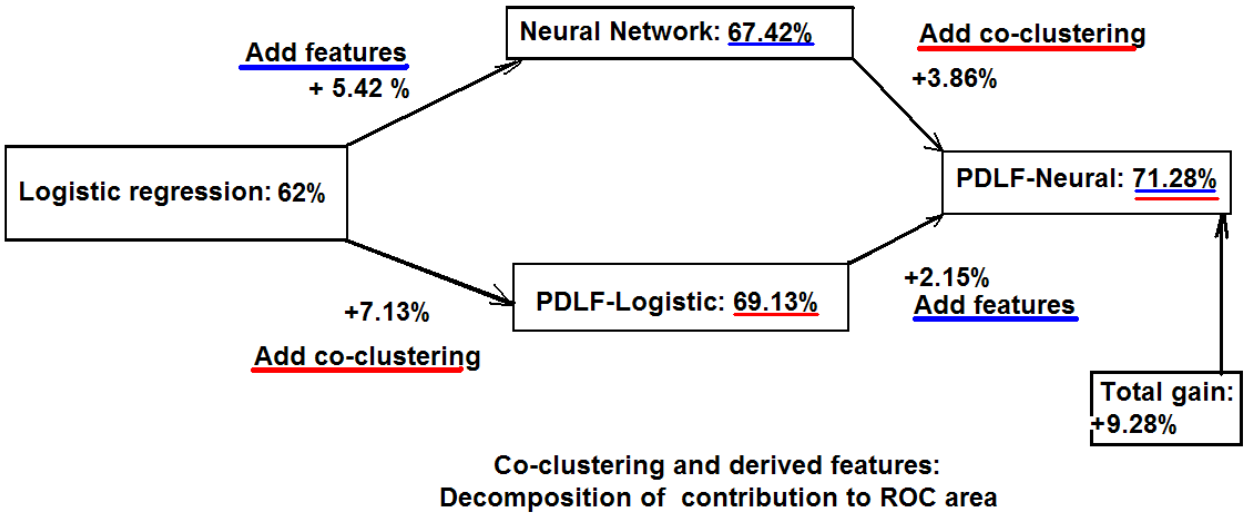
Title	Release date
Nosferatu (Nosferatu, eine Symphonie des Grauens) (1922)	1-Jan-22
Blue Angel, The (Blaue Engel, Der) (1930)	1-Jan-30
Pinocchio (1940)	1-Jan-40
Dial M for Murder (1954)	1-Jan-54
8 1/2 (1963)	1-Jan-63
Carrie (1976)	1-Jan-76
Top Gun (1986)	1-Jan-86
Bram Stoker's Dracula (1992)	1-Jan-92
Mortal Kombat: Annihilation (1997)	1-Jan-97
Sphere (1998)	13-Feb-98

There is nothing peculiar about the genres, except that there are no documentaries in the cluster and the most popular genres are comedy and drama. The user age ranges from 14 to 70, with 23 females and 55 males who hold 18 out of possible 20 occupations. All possible geographical locations are represented among the users in the cluster.

The identification of new covariates means the identification of significant interaction terms derived from the existing covariates. For instance, one could guess that there is an interaction between gender and a binary documentary covariate. If we introduce a new binary covariate that is equal to 1 for a male user rating a non-documentary then we could possibly improve the fit. Some more rigorous factor screening methods can be applied within the outstanding clusters.

6. Discussion

At first sight, the improvement of performance, ROC area, due to switching from logistic to neural model is small: 69.13% VS 71.28%. However, the following diagram shows that the gain in performance is actually dependent on the order in which the additional linear features (i.e., switching to neural model) and co-clustering variables are added to the baseline logistic regression model.



Co-clustering and derived features:
Decomposition of contribution to ROC area

When extra 39 linear features are added first, the gain in performance is larger than when they are added after co-clustering, and vice-versa: when co-clustering is applied first, the gain is larger than when it is applied second. The order and magnitudes of these gains suggest that the extra linear features are about as useful as the co-clustering, but the problem is that for this dataset the information derived from the covariates is correlated with that derived from the local structure. That is why the marginal gain of one, given that the other is already in the model, is reduced.

These findings show that, in general, the use of neural network can be justified because it is not guaranteed that the information in covariates is so much correlated with the information in the local structure that the simpler PDLF-GLM model is good enough.

7. Related work

An early empirical work related to PDLF is presented in [10]. The commonality with PDLF is that [10] does utilize both the local structure information and the demographic data in order to build a recommendation system. The local structure is handled by collaborative filtering approach, which was also utilized in MovieLens project. For instance, pairwise Pearson correlations are computed between the row-vectors of ratings from different users. For a fixed (user, movie) the rating is predicted as a sum of other ratings for that movie weighted by the correlation coefficients between the user of interest and other users who

already rated the movie. Pairwise correlations between the columns of ratings can be used similarly.

The demographic data are handled by a separate model and then the relative rankings from the two models are combined simply as a sum. Apparently, such model is very empirical and less powerful than PDLF where both approaches are tied together and have a certain theoretical justification.

Another method that can utilize both observed and latent information is clustering on k-partite graphs [8]. For instance, a tri-partite graph can consist of (Web page, Search query, Web user) triples. For a k-partite graphs, a so-called Relation Summary Network is constructed, which is supposed to expose the hidden structures. Like PDLF, the clustering algorithm utilizes Bregman divergences. If covariates are present, they are considered a new type of nodes. Thus, a k-partite graph with p covariates can be handled as a (k+p)-partite graph without covariates. In such case, clustering is performed w.r.t. both k-partite graph nodes and covariates and certain information is derived from covariates to improve the clustering on the original graph.

It appears that RSN method, as follows from its name, is aimed more at describing the hidden structures as opposed to predicting the response such as movie rating. For that purpose, PDLF appear more appropriate, even though currently not capable of working with k-partite data except for $k = 2$. Extending the method for $k > 2$ could be a good topic of future research.

Finally, the model of [4] works with similar dyadic data that consist of (Web page, Web ad) pairs. Unlike RSN model, this model learns the structure of dyadic data specifically for the purpose of response prediction (number of ad clicks), which relates it to PDLF. Because there are certain physical restrictions on the number of times an ad has to be displayed and the capacity of a web page, the problem of maximizing the overall expected click rate turns out to be that of linear programming. The coefficients for the linear model are derived from the EM-based clustering (not co-clustering, because only Web pages get clustered), which differs it from co-clustering in PDLF. Besides, no observed covariates are utilized.

Therefore, PDLF-based model(s) appear to be the most preferable when working with dyadic data for the purpose of dyadic response prediction.

References

[1] Agarwal, Merugu. Predictive Discrete Latent Factor Models for Large Scale Dyadic Data, *KDD'07*, August 12–15, 2007, San Jose, California, USA.

[2] A. Banerjee, I. Dhillon, J. Ghosh, S. Merugu, and D. Modha.
A generalized maximum entropy approach to Bregman co-clustering and matrix approximation. *JMLR*, 2007.

[3] D. Chakrabarti, S. Papadimitriou, D. Modha, and C. Faloutsos. Fully automatic cross-associations. In *KDD*, 2004.

[4] D. Chickering, D. Heckerman, C. Meek, J. C. Platt, and B. Thiesson. Targeted internet advertising using predictive clustering and linear programming.
http://research.microsoft.com/~meek/papers/goal-oriented.ps

[5] I. Dhillon, S. Mallela, and D. Modha. Information-theoretic co-clustering. In *KDD*, 2003.

[6] G. Golub and C. Loan. *Matrix Computations*. John Hopkins University Press, Baltimore, MD., 1989.

[7] D. L. Lee and S. Seung. Algorithms for non-negative matrix factorization. In *NIPS*, pages 556–562, 2001.

[8] B. Long, X. Wu, Z. Zhang, and P. S. Yu. Unsupervised learning on k-partite graphs. In *KDD*, 2006.

[9] MovieLens data set:
http://www.grouplens.org/node/73

[10] M. Pazzani. A framework for collaborative, content-based and demographic filtering. *Artificial Intelligence Review*,
(5-6):393–408, 1999.