Predicting labels for dyadic data

Aditya Menon and Charles Elkan, UCSD

Dyadic prediction and latent feature methods

Dyadic prediction is a problem superficially similar to supervised learning, except that our examples are dyads or pairs of objects:

Labelled dyads $((r_i, c_i), y_i) \rightarrow$ Predict y on an unseen dyad (r', c')

This problem can be modelled as a form of matrix completion, where our input is a matrix X whose rows are indexed by r_i, and columns by c_i. X contains entries only for the observed dyads (r_i, c_i), and our goal is to fill in the missing entries. Dyadic prediction has received a lot of attention of late, mostly in the context of collaborative filtering.

A popular strategy for dyadic prediction is learning latent features. Conceptually, we decompose the $m \times n$ matrix $X \approx UV^T$, where $U \in \mathbb{R}^{n \times k}$ and $V \in \mathbb{R}^{m \times k}$ Here, $k \ll \min(m, n)$ is the number of latent features. Training is performed via the following non-convex optimization:

$$\min_{U,V} ||X - UV^T||_O^2 + \frac{\lambda_U}{2}||U||_F^2 + \frac{\lambda_V}{2}||V||_F^2$$

where $||\cdot||_O^2$ is the Frobenius norm only on observed entries.

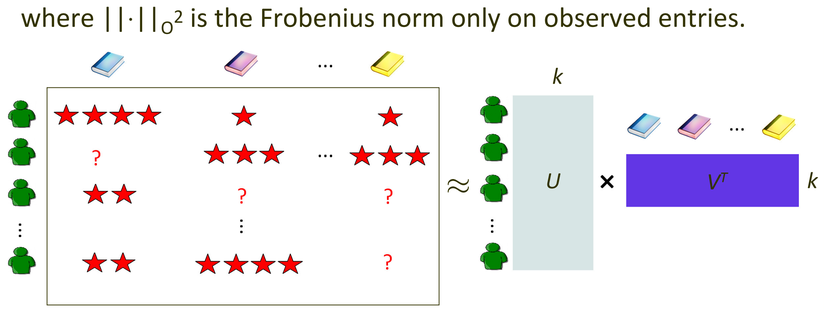

Figure 1. *Dyadic prediction via latent features for a collaborative filtering setting.*

A new problem: dyadic label prediction

We define a new problem, dyadic label prediction. Here, the input consists of the matrix X, and additionally a label matrix Y that comprises labels for the row/columns of X. The new task is:

Labelled dyads $((r_i, c_i), y_i)$ + Labelled objects $(r_i, y_i^r) \rightarrow$ Predict label on an unseen object r'

We allow Y to be a matrix so that we can handle multi-label prediction as well. This problem has important real-world applications:

- Will a user with known movie ratings respond to a new advertising campaign?
- Is a user in a social network "suspicious" by virtue of his contacts, some of whom are known to be suspicious?
- Will a bacterial strain appear in a food processing plant? [2]

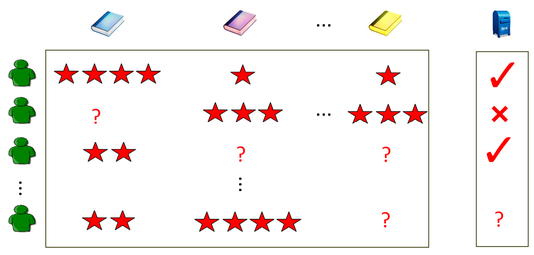

Figure 2. *Dyadic label prediction as matrix completion, collaborative filtering setting.*

A special case of dyadic label prediction is the problem of within-network classification, where the input is a graph with labels on a few nodes, and where the goal is to label all nodes. We emphasise that it is a genuine special case: with the dyadic interpretation, we can additionally allow for partially observed edges, and also seamlessly leverage additional information about the nodes, known as side-information.

Latent feature solutions

As noted before, dyadic label prediction is a generalization of supervised learning. Note that if we had explicit features U for the rows, then one can in fact reduce the problem to supervised learning: simply learn a weight matrix W e.g. via ridge regression:

$$\min_W ||Y - UW^T||_F^2 + \frac{\lambda_W}{2}||W||_F^2$$

In general we don't have such a U. But we can learn a U matrix using the latent feature approach! This solution strategy is essentially that followed by the SocDim method [3] for within-network classification on symmetric graphs. SocDim first transforms the data matrix to form the modularity $Q(X)$, which normalizes X using the vector of node degrees, d:

$$\mathcal{Q}(X) = X - \frac{1}{2|E|}dd^T$$

The latent features are taken to be the eigenvectors of $Q(X)$. (This assumes no missing data in X). One assumption this makes is that the same features that are predictive for reconstructing X are also predictive for Y. But that need not be strictly true in general. A natural extension of the SocDim idea is to jointly learn a U matrix to model both X and Y. From a collaborative filtering viewpoint, this can be thought of as treating the labels as movies!

The new objective is:

$$\min_{U,V,W} \left|\left| [X \quad Y] - U \begin{bmatrix} V \\ W \end{bmatrix}^T \right|\right|_O^2 + \frac{1}{2}(\lambda_U||U||_F^2 + \lambda_V||V||_F^2 + \lambda_W||W||_F^2)$$

This is a conceptually appealing reduction, but is it practically useful? Arguably not, because in most cases our final goal is accuracy on reconstructing Y, with reconstruction of X purely serving as a means to that end. Therefore, it is sensible to have a tradeoff parameter, μ, between the two reconstructions. Doing so yields the supervised matrix factorization (SMF) method [4].

$$\min_{U,V,W} \mu||X - UV^T||_O^2 + (1-\mu)||Y - UW^T||_O^2 + \frac{1}{2}(\lambda_U||U||_F^2 + \lambda_V||V||_F^2 + \lambda_W||W||_F^2)$$

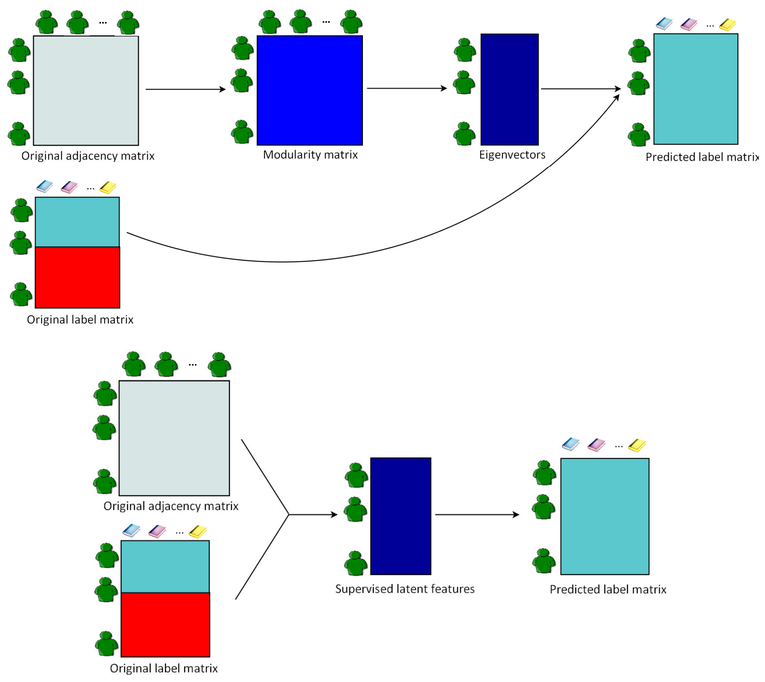

Figure 3. *Illustration of the SocDim (top) and SMF (bottom) methods.*

Using a latent feature log-linear model

At this stage, we can shift from the simple matrix factorization approach to more powerful dyadic prediction approaches. This brings two advantages. First, we can deal with missing entries in X. Second, we can allow for arbitrary missing data patterns in Y: unlike in standard multi-label learning, we can allow for cases where no row has a fully observed label..

We will look to use the latent feature log-linear model (LFL) of [1], which has further advantages:

- The ability to use side-information: e.g., to score suspiciousness, it is useful to know the user's age, gender and occupation.
- The prediction of well-calibrated probabilities over the labels. This is essential whenever a classifier is only part of a larger system.
- Handling both nominal and ordinal labels in the data matrix X.

For settings where the labels lie in the discrete range {1, …, R}, the LFL model is:

$$p(X_{ij} = r|U, V) = \frac{\exp((U_i^r)^T V_j^r)}{\sum_{r'} \exp((U_i^{r'})^T V_j^{r'})}$$

If we assume that X consists of ordinal values, the objective is:

$$\min_{U,V,W} ||X - \mathcal{E}(X)||_O^2 + \frac{1}{2}\left(\sum_r \lambda_U||U^r||_F^2 + \lambda_V||V^r||_F^2\right) + \sum_{(i,l)\in\mathcal{O}} \frac{e^{Y_{il}(W_l^T U_i)}}{1 + e^{W_l^T U_i}} + \frac{\lambda_W}{2}||W||_F^2$$

where $\mathcal{E}(X)_{ij} = \sum_r r p(X_{ij} = r|U, V)$.

Comparison of approaches

Below we compare the salient properties of the methods:

Item	SocDim	SMF	LFL
Supervised latent features?	No	Yes	Yes
Asymmetric graphs?	No	Yes	Yes
Missing data?	No	No	Yes
Latent features of?	Modularity	Data	Data
Single minimum?	Yes	No	No

Table 1. *Comparison of the various approaches.*

We observe a similarity in the methods' objective functions. Take the special case of a symmetric graph with no missing edges, and let the tradeoff parameter $\mu = 0$. Then, for appropriate functions f and g, all methods optimize:

$$\min_{U,\Lambda} ||f(X) - g(U, \Lambda)||_F^2 + \frac{\lambda_U}{2}||U||_F^2 + \frac{\lambda_\Lambda}{2}||\Lambda||_F^2$$

Specifically, the objective functions are:

SocDim $\min_{U,\Lambda \text{ diagonal}} ||\mathcal{Q}(X) - U\Lambda U^T||_F^2$

SMF $\min_U ||X - U\Lambda U^T||_F^2 + \frac{\lambda_U}{2}||U||_F^2$

LFL $\min_U ||X - \sigma(UU^T)||_O^2 + \frac{\lambda_U}{2}||U||_F^2$

We see that SocDim and LFL are both based on the idea of transformation, but of different things: SocDim transforms the raw data matrix, while LFL transforms the estimate matrix.

Transforming the data matrix has analogues in spectral clustering, where the graph Laplacian is used to normalize the effect of degrees on nodes before they are clustered. Transforming the estimate matrix, on the other hand, is simply to ensure valid predictions in the range [0, 1].

If we compare SocDim and SMF, then we see the optimizations are essentially the same, apart from the issue of the modularity transform in SocDim. However, note that SocDim is immune to local optima, as it relies on a closed form eigenvector computation. This is a potentially very important point in its favour.

Experiments

There are several questions we wish to study empirically:

- Do supervised latent features help? In particular, does their nonconvexity in training offset their predictive power when compared to the unsupervised SocDim method?
- Does the choice of data transform matter? Can we use the Laplacian with SocDim? Does using the Laplacian/Modularity improve the performance of SMF?
- If we use a naïve scheme for dealing with missing entries, will SocDim perform competitively on partially observed datasets?

We used three datasets in our experiments:

- `blogcatalog`, comprising fully-observed links between 2500 bloggers from the BlogCatalog directory. Labels are the user's interests divided into 39 categories i.e. this is a multi-label task.
- `senator`, consisting of the Yea/Nay votes of 101 senators for 315 bills in the 109th house of the senate (with only a few missing entries). The label is if the senator is a Republican or Democrat.
- `usps`, consisting of handwritten digits represented as grayscale 16 x 16 images. For simplicity, the images are all binarized so that the pixel values are either { 0, 1 }. We occlude some pixels so that X has missing entries. The labels are the images' true digits.

To apply SocDim and SMF on the `usps` dataset, we merely treated missing pixels as having the value 0 (which is the dominant pixel value after binarizing the images).

We used the 0-1 error measure for the `senator` and `usps` datasets. For the multilabel `blogcatalog` dataset, we used the F1-micro and F1-macro scores, which for a true label matrix y and predicted matrix $\hat{y}$ are:

$$\text{Micro} = 2\frac{\sum_{i,l} y_{il}\hat{y}_{il}}{\sum_{i,l} y_{il} + \hat{y}_{il}} \qquad \text{Macro} = \frac{2}{L}\sum_l \frac{\sum_i y_{il}\hat{y}_{il}}{\sum_i y_{il} + \hat{y}_{il}}$$

The scores lie in [0, 1], with higher values being better.

The results are presented below. We show both train and test error for three choices of data transformation: none, modularity, and Laplacian.

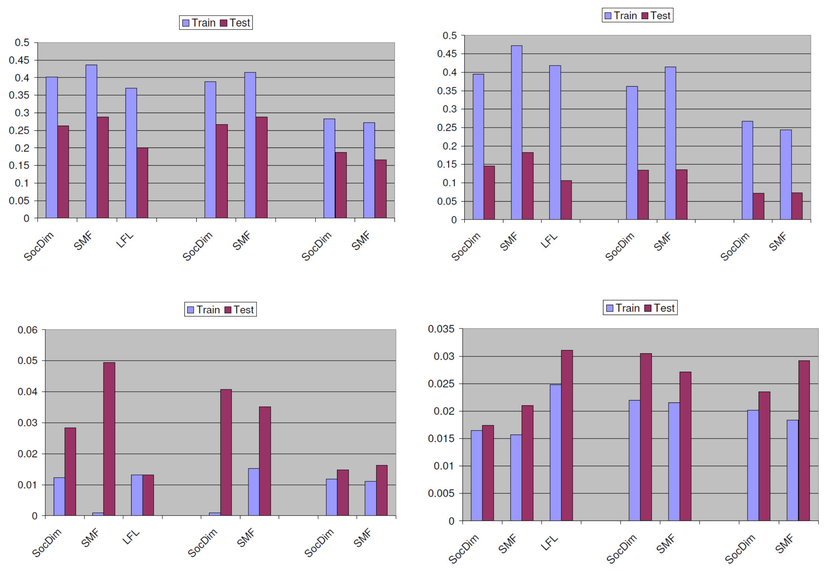

Figure 4. ***Top row:*** *Results on* `blogcatalog` *dataset, Micro and Macro measures.* ***Bottom row:*** Results on `senator` *and* `usps` *datasets.*

We see that SocDim performs well in general, suggesting that its immunity to local optima is an important feature that offsetts its unsupervised features. In contrast, the SMF and LFL methods sometimes suffer from overfitting (see `blogcatalog` results).

The choice of data transformation appears to make only a mild difference – interestingly, the raw data matrix without any transformation often gives quite competitive results for SocDim.

The good performance of SocDim on `usps` suggests that even a naïve scheme for dealing with missing data may be good enough for it to learn useful latent features. This warrants a careful study of the situations where SocDim may be similarly applied.

References

[1] Aditya Krishna Menon and Charles Elkan. Dyadic prediction using a latent feature log-linear model. http://arxiv.org/abs/1006.2156, 2010.
[2] Purnamrita Sarkar, Lujie Chen, and Artur Dubrawski. Dynamic network model for predicting occurrences of salmonella at food facilities, In BioSecure Intl. Workshop, 2008.
[3] Lei Tang and Huan Liu. Relational learning via latent social dimensions, SIGKDD 2009.
[4] Shenghuo Zhu, Kai Yu, Yun Chi, and Yihong Gong. Combining content and link for classication using matrix factorization, SIGIR 2007.