Final version - information reduction algorithm (Deliverables D4.1, D4.2)

E. Gutflaish (M.Sc. student); Prof. A. Kontorovich (PI); Dr. S. Sabato (PI)

5th June 2017

Contents

1 General idea

I will refer to *DB_USER* as the main object and *TABLE* as the feature field but any other fields, a combination or Cartesian product of fields, can fill these parts. the measure of relevance of a db-table to a db-user ("db" is short for "database"). Let n, m, k be the number of db-users, number of db-tables and the output rank, respectively. We will have a learning stage on off line data, which is supposed to be free of anomalies (or at least mostly). Then our classifier will be ready to use for a batch of test intervals, in which the classifier would rank these intervals according to their anomaly score. Note that all of the following procedures are implemented in MATLAB.

For more details on our approach please refer to arXiv - Temporal anomaly detection: calibrating the surprise

1.1 General guide lines

Let n and m be the number of db-users and number of db-tables, that are seen during the low rank approximation stage respectively. Let u_i and t_j denote db-user i and db-table j, respectively. Let B_t represent a binary matrix so that $B_t(i, j) = 1$ if user u_i accessed table t_j on the time t, else $B_t(i, j) = 0$. Let T_1, T_2 and T_3 be the amount of intervals provided for the low rank approximation, regression stage and test stage respectively. Any matrix $B_t \in \mathbb{R}^{n_t, m_t}$ after the low rank approximation will in the the following formation:

$$\begin{bmatrix} B_t(1,1) & \dots & B_t(1,m) & \dots & B_t(1,m_t) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ B_t(n,1) & \dots & B_t(n,m) & \dots & B_t(n,m_t) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ B_t(n_t,1) & \dots & \dots & \dots & B_t(n,m_t) \end{bmatrix}$$

hence, any new matrix will be assembled from seen users and tables, while the new ones comes after them in the access matrix.

The IBM regression is autoregressive with a 24-interval window, i.e. demands 24 intervals to serve as pivots. We assume that those intervals are free of anomalies. This is required for the regression stage and the classification stage. The Classification stage will return anomaly scores on intervals 25+.

1.2 Software Requirements

- Matlab - version $\geq 2016b$
- Matlab parallel toolbox is not required but is strongly recommended for utilizing multi cores

2 Learning and Classification

Now we will describe the learning and classification scenarios. The software package also includes a *Demo.m* script that show the full learning and classification scenarios. Additionally the matrices that correspond to the 2 month IBM dataset we received is supplied in Folder *DateSet*, based on the fields $[DB - User, DB - table]$.

2.1 Learning stage

In the learning stage we get as input:

- *mat_arr_cell_train_model* - MATLAB cell array of $B_t \in \mathbb{R}^{n,m}$ $\forall t = 1, \dots, T_1$
- *mat_arr_cell_train_reg* - MATLAB cell array of $B_t \in \mathbb{R}^{n_t,m_t}$ $\forall t = 1, \dots, T_2$
- *files_mat_reg* - MATLAB cell array of $\{strings \in [year, month, day, hour], |year| = 4, |month| = |day| = |hour| = 2 \ \forall t = 1, \dots, T_2\}$

The is done by executing the procedure $LearnModel(mat\_arr\_cell\_train\_model, mat\_arr\_cell\_train\_reg, files\_mat\_reg)$ output:

- Folder - IBM_New_Model
  - Folder - $Model$
    - $U \in \mathbb{R}^{n,k}$
    - $S \in \mathbb{R}^{k,k}$
    - $V \in \mathbb{R}^{m,k}$
    - $data \in \mathbb{R}^{n,m}$ - Original users and tables based on *mat_arr_train_model*
  - Folder - $reg/regIBMUnseen$
    - $reg\_final\_results$ - cell array
      - W - Regression coefficients

2.2 Testing a batch of of intervals

In the Classification stage we get as input:

- *mats_test* - MATLAB cell array of $B_t \in \mathbb{R}^{n_t,m_t}$ $\forall t = 1, \dots, T_3$
- $file\_mats\_names\_test$ - MATLAB cell array of $\{strings \in [year, month, day, hour], |year| = 4, |month| = |day| = |hour| = 2 \ \forall t = 1, \dots, T_3\}$

The is done by executing the procedure $percent\_dist\_items = getTestRanking(mats\_test, file\_mats\_names\_test)$ output:

- $percent_d ist_i tems \in \mathbb{R}^{T_3}$, which contains the anomaly score in % for every interval.

3 Future Research

Our approach of finding the low rank model is based on an hybrid optimization between the squared loss and the likelihood loss. this is due to the hardness of optimizing with respect to the likelihood with a Bernoulli model, since the optimization demands bounded values $\in [0, 1]$ on the log loss. Additionally the log loss is *sensitive* around 0, which creates further numerical issues. These issue are usually resolved in the literature by assuming a simple exponential model (e.g. a Gaussian), but our purpose is to solve this problem in a Bernoulli based model. We are currently working on solving these issues and solve the optimization with respect to the log loss instead of a the squared loss.

4 Other possible variations of use of the software package

Since we produce a "low rank" approximations for users and tables which is the probability access matrix. One might want to utilize this structure for other purposes, e.g. clustering or a 3d visualization of the users (or tables) or even to Use the probabilities in some manner. The following procedure projects users onto their latent representation. The input is a binary row vector indicating which tables the user accessed.

Notice that only previously seen tables are considered in the projection of the user.

Algorithm 1 projectToLatentUser(u, V)

```
  **Input**: $u \in \mathbb{R}^{\mathbf{1,m}}$, $V \in \mathbb{R}^{\mathbf{m,k}}$
  **Output**: $u_{new} \in \mathbb{R}^{\mathbf{k}}$

  **return** $u \cdot V$
```

Similarly, a method $projectToLatentUser(t, U)$ for tables exists.