HAL
open science

Incremental learning with social media data to predict near real-time events

Duc Kinh Le Tran, Cécile Bothorel, Pascal Cheung Mon Chan, Yvon Kermarrec

To cite this version:

Duc Kinh Le Tran, Cécile Bothorel, Pascal Cheung Mon Chan, Yvon Kermarrec. Incremental learning with social media data to predict near real-time events. DS 2014 : 17th International Conference on Discovery Science, Oct 2014, Bled, Slovenia. pp.180 - 191, 10.1007/978-3-319-11812-3_16 . hal-01185078

HAL Id: hal-01185078

https://hal.science/hal-01185078v1

Submitted on 19 Aug 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Incremental learning with social media data to predict near real-time events

Duc Kinh Le Tran[1,2], Cécile Bothorel[1],
Pascal Cheung Mon Chan[2], and Yvon Kermarrec[1]

[1] UMR CNRS 3192 Lab-STICC
Département LUSSI – Télécom Bretagne
{duc.letran, cecile.bothorel, yvon.kermarrec}@telecom-bretagne.eu
[2] Orange Labs
{duckinh.letran, pascal.cheungmonchan}@orange.com

Abstract. In this paper, we focus on the problem of predicting some particular user activities in social media. Our challenge is to consider real events such as message posting to friends or forwarding received ones, connecting to new friends, and provide near real-time prediction of new events. Our approach is based on *latent factor models* which can exploit simultaneously the timestamped interaction information among users and their posted content information. We propose a simple strategy to learn incrementally the latent factors at each time step. Our method takes only recent data to update latent factor models and thus can reduce computational cost. Experiments on a real dataset collected from Twitter show that our method can achieve performances that are comparable with other state-of-the-art non-incremental techniques.

Keywords: social media mining, incremental learning, latent factor models, matrix factorization

1 Introduction

Recent years have witnessed the explosion of social media on the Internet. Vast amounts of user-generated content are created on social media sites every day. Social media data are often characterized as vast, noisy, distributed, unstructured and dynamic [7]. These characteristics make it difficult or impossible to apply conventional data mining techniques on social media data.

One of the challenges in mining social media is how to leverage the *interaction information* (or *relation*) in the data. Interaction information in social media can be any type of interactions between two users (e.g send a message, write a comment) or relations between them (e.g friendship declared in a social network). These interactions and relations are heterogeneous (can be different in nature) and very rich in volume. In general, interaction information is worthy to consider. Conventional machine learning techniques relying on attribute-value data representation (i.e *content information*) cannot fully exploit this kind of information.

Another challenge of mining social media data lies in the fact that these data are vast and continuously evolving. Social media provide a continuous stream of data. Some applications in social media mining require building prediction model to periodically extract useful information. Using offline learning techniques (*batch learning*), we have to consider all data available from the past until the present. This approach is not suitable for mining social media because (1) as new data come, the size of the dataset grows, it gets more and more expensive to learn and to apply the model (2) this approach treats old data in the past and recent data the same way; intuitively, this may not be a good choice because old data often contain less relevant information (in the context of predicting future events).

These two challenges have not often been considered together. Recently, there have been some works on mining social media stream, for example [1, 9], but they mostly concern topic extraction or trending topic detection on social media. We are interested in predicting actions or attributes on each user. For this problem, there have been a lot of works on exploring relational information in data. These techniques are often referred to as *statistic relational learning* [5]. Unfortunately they can only deal with static datasets. On the other hand, the second challenge can be overcome by using *incremental learning* techniques [8], which are capable of incrementally updating the model with new data. However, most incremental learning algorithms only deal with attribute-value data.

This paper aims to tackle both these two challenges. We are interested in predicting some particular users' actions in social media: post a message mentioning a telecommunication brand on Twitter. The problem is described in details in the next sections. We show that our proposed method based on *latent factor models* achieves comparable or better performances than other learning techniques in leveraging simultaneously interaction information and attribute-value information in social media. The basic idea of our method has been introduced in our previous work [14], but here we test it for a different task and in a different context. We also show that incremental learning is more appropriate for mining social media: it is at least as good as batch learning in terms of prediction performance and can gain a lot in computational time.

2 Data representation and problem statement

For reasons of convenience, we adopt the concept of the *social attribute network (SAN)* [6] to represent the data from social media. A *SAN* is a social network G_s=(V_s, E_s) where V_s is the set of nodes and E_s is the set of (undirected) edges. The social graph is augmented with a bipartite graph $G_a = (V_s \cup V_a, E_a)$ connecting the *social nodes* in V_s with *attribute nodes* in V_a. The edges in E_s are *social links* and the edges in E_a (connecting social nodes and attribute nodes) are *attribute links.* The value of an attribute a for a social node u is represented by the weight of the link (u, a). Social media data can be represented by a SAN as follows: social nodes represent the users, social links represent their relations

or interactions and attribute links represent known values of attributes (profiles, user-generated content) on these users.

The advantage of the SAN model is that it can easily represent data stream from social media. When data come in stream, at each new time step, we can have new users and new attributes. We can also have new social relations and interactions between users and new values of attributes for each user. All of these elements can be represented by an "incremental" SAN: at each time step new nodes (social nodes or attribute nodes) and new links (social links or attribute links) are added. The new links can be social links or attribute link and concern both existing nodes and new nodes. To be clear, we do not consider node and link disappearance.

Our objective is to predict a target variable on the users in the next time step $t+1$ using data up to (including) t. In this paper, we consider a binary target variable (label) which concerns some particular real-time action of the users. In each time step, it takes the value 1 (positive) if the user take the action and 0 (negative) otherwise. This is a near real-time prediction problem (i.e requires building prediction model at each time step). We aim to design prediction models that can be learned incrementally from an "incremental" SAN . Our problem is, *with new nodes and new links added at each time step, how to adapt the model built at the previous time step to get a new model.*

3 Related work

The problem stated above is concerned with building classifiers from both attribute-value data and the social graph. To use attribute-value data, any conventional machine learning technique can be employed. Among these techniques, *support vector machine* (SVM) [4] is one of the most robust.

To explore the social graph, many techniques of statistical relational learning have been proposed. We cite here some interesting graph-based approaches. The neighbor-based approach [12] infers the target attribute of a node from that of its neighbors as follows:

$$P(y_i = 1) = \frac{\sum_{j \in N_i} y_j S_{ij}}{\sum_{j \in N_i} S_{ij}} \tag{1}$$

where y_i denotes the attribute value of the node t (0, 1 where 1 corresponds to a positive label), S_{ij} denotes the weight of the social link (i, j) and N_i is the set of neighbors of t. This is a very simple approach it was proven to be better than other relational techniques in some particular datasets [12].

Another approach of using the social graph for classification is *Social Dimension* [13]. The basic idea of this method is to transform the social network into features of nodes using a graph clustering algorithm (where each cluster, also called a *social dimension*, corresponds to a feature) and then train a discriminative classifier (SVM) using these features. Any graph clustering algorithm can be used to extract social dimensions but *spectral clustering* [11] was shown to be

the best. This approach helps exploiting the graph globally, not just the neighborhood of a node. It was shown in [13] that the *Social Dimension* outperforms other well-known methods of graph-based classification on many social media datasets.

There have also been efforts to use both the social graph and the attributes to improve prediction performances. For example, the *Social Dimension* approach [13] was extended to handle attribute on nodes. It is a simple combination of features extracted from the social graphs with attributes on nodes to learn a SVM classifier.

Another class of interesting techniques for mining both the social graph and the attributes are based on *latent factor models*(*LFM*) [2]. LFMs represent data points (in this case social nodes and attribute nodes) as vectors of unobserved variables (latent factors or latent features). All observation on nodes (in this case, links between nodes) depend on their latent factors. When we only have attribute-value data, latents factors can be learned using *matrix factorization (MF)* techniques, which consists of decomposing the data matrix into two matrices: one contains latent features and one contains those of attributes. To use the relational information in the social graph, in [10] the authors proposed an extension of MF, called *relation regularized matrix factorization* (*RRMF*). RRMF simultaneously exploits the social graph and the attribute graph. Suppose that we have a dataset represented by a SAN $\mathcal{G}$, RRMF learns latent factors by minimizing:

$$\begin{aligned} Q\left(U, P, \mathcal{G}\right) = & \alpha \sum_{(i,j)\in E_s} S_{ij} \left\|u_i - u_j\right\|^2 + \sum_{(i,k)\in E_a} \left(A_{ik} - u_i p_k^T\right)^2 \\ & + \lambda \left(\sum_{i=1}^{n_s} \left\|u_i\right\|^2 + \sum_{k=1}^{n_a} \left\|p_k\right\|^2 \right) \end{aligned} \tag{2}$$

where E_s is the set of social links, E_a is the set of attribute links; S is the adjacent matrix of the social graph and A is the adjacent matrix of the bipartite attribute graph; U is the matrix constituted of the latent vectors of all the social nodes and similarly, P is the matrix constituted of the latent vectors of all the attribute nodes of $\mathcal{G}$. The parameter α allows to adjust the relative importance of the social network in the model. The third term is a regularization term to penalize complex models with large magnitudes of latent vectors. λ is a regularization parameter. We can see that this is in fact the factorization of the attribute matrix A when adding regularization term $\alpha \sum_{(i,j)\in E_s} S_{ij} \left\|u_i - u_j\right\|^2$. This term is called the *relational regularization term* which allows to minimize the distances between connected social nodes in the latent space. The RRMF approach assumes that connected social actors tend to have similar profiles. In some cases, it is better to use the *normalized Laplacian* of the social graph and the regularization term becomes $\alpha \sum_{(i,j)\in E_s} S_{ij} \left\|u_i/\sqrt{d_i} - u_j/\sqrt{d_j}\right\|^2$ where d_i is the degree of the node i in the social graph (see [10] for more details). The

latent factors learned with RRMF are then used to train a classifier for label prediction problem.

All techniques mentioned above concern batch learning, i.e learning from a static dataset. In their problem setting, they assume that the set of social nodes is partially labeled and the problem is to infer labels of unlabeled nodes. Our problem (near real-time prediction problem) is not in same context but we can use the same idea : using latent factors to train a classifier at each time step. In the next sections, we describe our method in which we learn LFM (more precisely RRMF) incrementally and then use these factors to predict labels at each time step. We describe our strategy (based on least squares regularization) and show that it works well in a real world problem with data collected via Twitter.

4 Incremental learning with latent factor model

In the incremental learning context defined in Section 2, we need to learn a model (i.e the latent features of nodes) at each time step. The *batch learning* approach suggests that we learn the latent features at each time step using the whole snapshot of the SAN $\mathcal{G}(t)$ (which contains all nodes and links collected up to t)

$$U^{\star}(t),\, P^{\star}(t) = \underset{U,\, P}{\arg\min}\, Q\left(U,\, P,\, \mathcal{G}(t)\right) \tag{3}$$

where Q is the objective function defined above (Equation 2).

The incremental method learns a model (latent factors of nodes) only from new data (i.e the incremental part of the SAN, denoted by SAN $\Delta\mathcal{G}(t)$) when reusing the old model, i.e latent features of nodes calculated in the previous time step. To do this, we minimize the following objective function:

$$\begin{aligned} Q_{inc}(U,\, P,\, t) =& Q\left(U,\, P,\, \Delta\mathcal{G}(t)\right) \\ &+\mu\left(\sum_{i\in V_s(t-1)} \left\|u_i - u_i^{\star}(t-1)\right\|^2 + \sum_{k\in V_a(t-1)} \left\|p_k - p_k^{\star}(t-1)\right\|^2\right) \end{aligned} \tag{4}$$

where $V_s(t-1)$ and $V_a(t-1)$ are respectively the set of social nodes and the set of attribute nodes in the previous time step; $u_i^{\star}(t-1)$ and $p_k^{\star}(t-1)$ are respectively the latent vectors of the social node i and the attribute node k learned in the previous time step and μ is a parameter of the model. This objective function consists of two terms. The first term is the objective function of MF on the incremental graph $\Delta\mathcal{G}(t)$. The second term is a regularization term for minimizing the shifts of latent features of the same nodes between time steps. By minimizing the two terms simultaneously, we learn latent features of nodes both from the new data and from the latent features of existing nodes of the previous time step. We can easily see that the latent features of an existing node are updated if and only if there are new links connecting to it. With the second regularization term, we ensure that the latent space does not change much from

a time step to the next. The parameter μ allows to tune the contribution of the previous model to the current model.

After having calculated latent factors of nodes at the time step t, we have a low-dimensional representation of the data points (nodes) at this time step. It can be used for any standard machine learning task on the nodes. For our prediction problem, we are based on the hypothesis that data collected up to t are informative for the target variable in $t+1$ on the social nodes. With this hypothesis, we can use these factors (low dimensional representation of the data up to t) of the social nodes to train a classifier and then use the classifier to predict in the next time step.

In terms of optimization, we can use standard algorithms (e.g gradient-based) to minimize Q in Equation 3 for batch learning or Q_{inc} in Equation 4 for incremental learning. In this work, we adapt the *Alternating Least Squared (ALS)* algorithm [15]. The basic idea of this algorithm is to solve the least square problem with respect to the latent features of one node at a time (when fixing those of the others) until convergence. The complexity of the algorithm linearly depends on the number of squared terms in the objective function, which is the total number of nodes and number of links in the SAN. In other words, the learning algorithm has linear complexity with respect to the size of the data. In case of incremental learning, when optimizing only on recent data ($\Delta\mathcal{G}(t)$), we can gain a lot in terms of computational cost.

5 Experiments

5.1 Data description and experimental setup

The dataset was collected via Twitter API[1] in the period from July to December 2012. The data concern the followers of the Sosh account on Twitter (@Sosh_fr[2]) in this period. We keep the identities of the followers of @Sosh_fr in our database. During the period, new followers of @Sosh_fr were constantly added. For each follower, we regularly get the following elements: all the tweets, all the retweets, the list of followers (from which we can build the who-follow-whom graph among these users). We also collected some elements of the profile of each user (e.g. some variables related to the global centrality of the user such as the number of followers, the number of tweets posted, etc.).

We collected the data regularly enough to be able to build 20 week-based snapshots (the first week begins on 15/07/2012) of the dataset. We have totally 30 400 users, about 9×10^5 who-follow-whom links, about 36×10^4 tweets and 26×10^4 retweets (on average 18×10^3 tweets and 13×10^3 retweets per week). We want to use both the social interactions (follower-followee relation, retweet) and the tweets of users. We represent each snapshots by a SAN, the SAN for the week t is built as follows:

[1] https://dev.twitter.com/docs/api

[2] Sosh is a French mobile brand, developed in France by the French operator Orange since 6 October 2011. Sosh is on Twitter at http://twitter.com/Sosh_fr.

- *The social graph.* We put a social link between two users if they are linked (one follows the other) or if they have retweeted a common tweet in the period t. We can see that this is a sum of follower graph and *co-retweet* graph (we aggregate these two graphs to get a denser graph).
- *The attributes.* We consider each word in the tweet(s) of users as an attribute. We put an attribute link between a user and a word if the user posted a tweet containing the word in the period. The link is weighted by how many times this occurred.

We are interested in predicting who will talk about the brand Sosh (i.e mention @Sosh_fr, write the word "sosh" in their tweets) in the week $t+1$ using data up to (including) the week t. Among the followers of @Sosh_fr, the ones who will talk about "sosh" are often customers of Sosh or just people interested in the brand who could become the future customers of the brand. At each time step (week), the prediction problem is a classification problem where positive labels correspond to who talk about "sosh" in the next week. Figure 1 presents the number of users and number of positive labels in each week. The portion of positive labels in each week is relatively small (less than 1%).

We apply our method (*incremental LFM*) for this prediction problem. For each time step, we calculate latent factors of all nodes by minimizing the objective function defined in Equation 4. At the time step t, we use the latent factors to learn an SVM classifier with positive labels in the next time step $t+1$. At $t+1$, we use the model learned in the previous time step to predict positive labels in the next time step $t+2$. We use *Area Under ROC Curve* (AUC) [3] to measure the prediction performance. AUC is a rank measure which allows to measure prediction performance across all possible cut-off thresholds. Roughly speaking, it is the probability that a classifier will rank a randomly chosen positive instance higher than a randomly chosen negative one.

The number of latent factors D is set to 20. In the objective function we use the *normalized Laplacian regularization* term as we see that it achieved better performance than normal graph regularization. We fixed the number of iterations in our ALS optimization problem to 20 since we observed no improvement of performance beyond 20 iterations. The regularization parameter λ is set to 50; λ is set to 100 and μ is set to 100. The influences of these parameters are studied in Subsection 5.5.

5.2 Baselines

At each time step (week) t we apply the following baseline techniques to compare with our incremental method:

Trivial solution 1 Since a user can talk about "sosh" more than once, it is interesting to know if this is a repeated action: if a user have talked about "sosh", how likely will she/he talk about it again. This is the idea of this first baseline: who ever had a positive label (at least once) in the past will have a positive label the next week $t+1$.

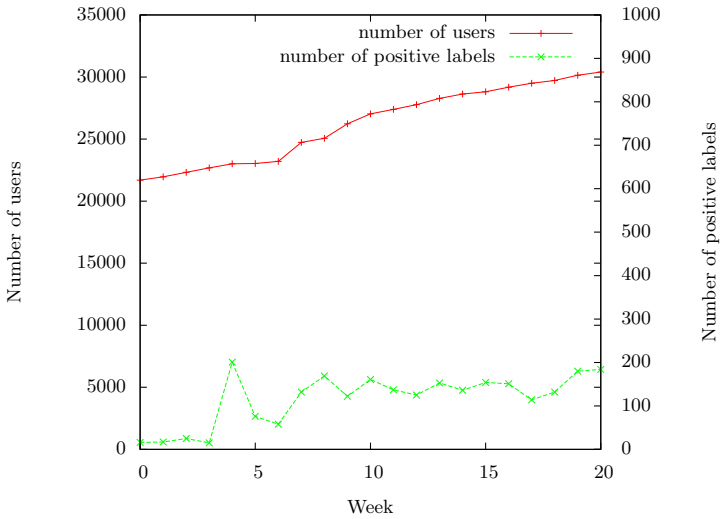


Fig. 1: Number of users and number of users having positive labels in each time step

Trivial solution 2 In Twitter, users have different levels of usage. There are users who write a lot of tweets, have lots of friends or followers, etc. These users are more likely to talk about "sosh". From this observation, we build a prediction method that predicts the label of a user based on how active she/he is in Twitter. We tried different measures of "activeness" of users, but we see that the number of tweets posted in the past is the best for this prediction problem.

Neighbor-based method This method use the neighborhood of each user in the social graph (described in Section 3). This method does not require a learning step, the label of a node in $t+1$ is inferred from that of its neighbors in the previous time step t (Equation 1).

Social dimension This method uses the social graph. At each time step we extract the social dimensions (described in Section 3) and then use these dimensions to train an SVM classifier with positive labels of the next time step (same procedure as with the latent factors in our method). The number of dimensions is set to 10. We do not see any improvement of performance setting this value bigger than 10.

SVM on attributes We use supervised classification with the attributes. At the time step t, we train an SVM classifier from all known values of attributes up to t and positive labels in $t+1$. Because the attributes are words (in the tweets), this means that we use the "bag of words" produced by each user up to t.

Social dimensions + attributes This is a combination of the *social dimensions* and the attributes. We use supervised classification (SVM) with the social dimension and the attributes.

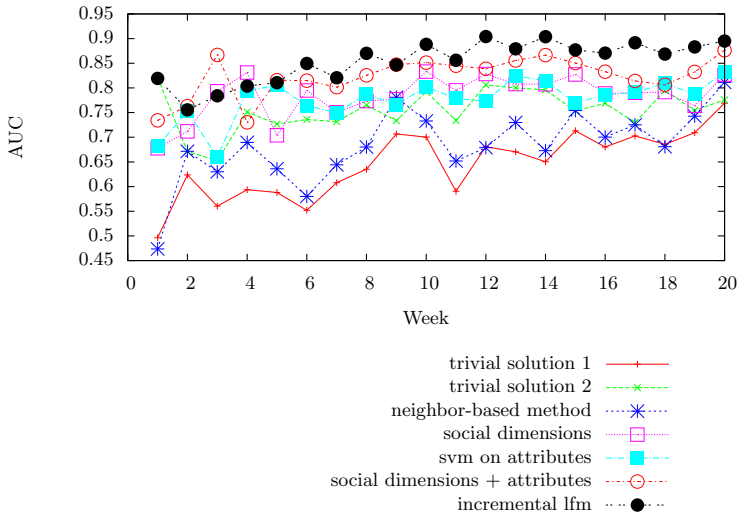


Fig. 2: Performances of different methods

5.3 Performance

Figure 2 presents the performances of all learning techniques. First of all, we can see that both the social graph and the attributes (bag of words for each user) are informative. The methods using these data (non-trivial) are often better than the trivial method. The neighbor-based method is not well adapted to this dataset : it gives even worse performance than the trivial solution 2. Combining these two sources gives even better performances (our method and Social Dimension+attributes).

Except for some perturbations in the beginning, our method (incremental LFM) achieves the highest AUC in all time steps. We conclude that, by exploiting both the social graph and the attributes we can enhance significantly the prediction performance and our incremental method based on LFM achieves relatively good performance in comparison with the best batch-learning technique.

5.4 Gain in computational time

Figure 3 shows computational times of the incremental LFM method and by the best baseline method - a combination of Social Dimension and attributes.

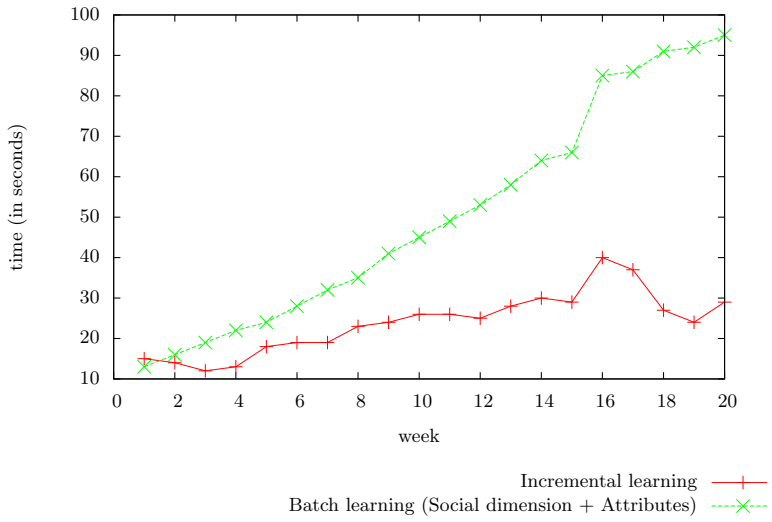
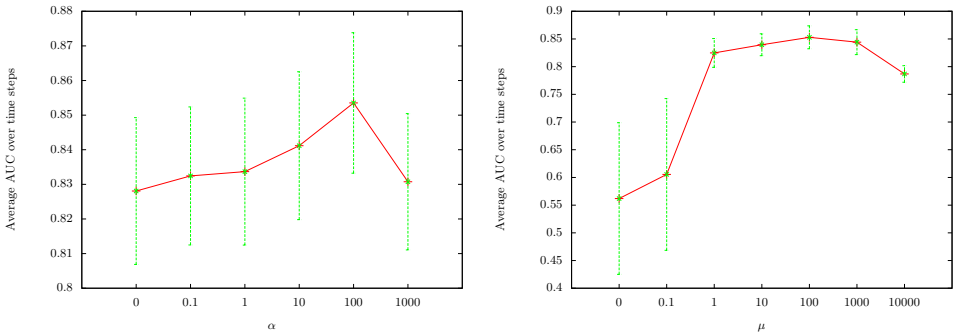


Fig. 3: Learning time

For our incremental method, computational time at each time step consists of learning latent factors (optimization) and training an SVM classifier. For the other methods, computational time consists of spectral clustering of the social graph and training an SVM classifier. We measure only the learning time (i.e optimization). To be fair, the two methods are implemented and executed on the same machine (Linux 64bit, CPU 8x2.1GHz). The figure shows significant gain in time using incremental learning. This is an illustration of our theoretical analyses in previous sections: learning with aggregated data becomes more and more expensive as new data are added; incremental learning only requires time to deal with new data.

5.5 Sensitivity to parameters

We examine the sensibility of 3 important parameters of our incremental LFM method : α, μ and the number of latent factors D. We average the performance (AUC) of all time steps to get a global performance for each parameter configuration. As shown in Figure 4a, too small or too large values of the parameter α hurt the performance. Larger α means that the social interactions have more contribution to the prediction model. When $\alpha = 0$, no interaction information is used. Maximum AUC is achieved around $\alpha = 100$. The effect of with the parameter μ is shown in Figure 4b. This parameter controls the contribution of the prediction model learned in the previous time step to the current model. $\mu = 0$ corresponds to the case where we learn latent factors only from recent data and latent factors learned in previous time step are not used. We see that when μ increases, the performance increases and achieves its maximum at $\mu = 100$. About



(a) α (when $D = 10$ and $\mu = 100$) (b) μ (when $D = 10$ and $\alpha = 100$)

Fig. 4: Sensitivity to parameters of the incremental LFM method

the number of latent factors D, we observe a small influence of this parameter on the performance. We see that small values of D are adequate because we can not improve significantly the performance setting it bigger than 10.

6 Conclusion

We have proposed an incremental learning method based on *latent factor model* for a prediction problem with data collected from Twitter. Our strategy (adding a regularization term) for incremental learning leads to very promising experimental results in both performance and computational cost. The main limitation of our method is how to choose the right values of its parameters to achieve its best performance. In future work, we will consider automatic configuration for the parameters at each time step to improve the performance. We plan on extended tests on other datasets or synthetic data to understand deeply the nature of the data where the method is efficient and robust. We keep working on the Twitter dataset but for other prediction problems (other type of events), the most interesting problem is to predict whether a user talks positively or negatively about the brand. We also consider other possible extensions of our models to handle more complicated data structure from social media: there are more than one types of social links in the SAN, directed links between social nodes, link disappearance etc.

References

1. L.M. Aiello, G. Petkos, C. Martin, D. Corney, S. Papadopoulos, R. Skraba, A. Goker, I. Kompatsiaris, and A. Jaimes. Sensing trending topics in twitter. *Multimedia, IEEE Transactions on*, 15(6):1268–1282, Oct 2013.

2. D J Bartholomew, M Knott, and I Moustaki. *Latent Variable Models and Factor Analysis: A Unified Approach*. Wiley Series in Probability and Statistics. Wiley, 2011.
3. Andrew P Bradley. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recogn.*, 30(7):1145–1159, 1997.
4. Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.
5. L Getoor and B Taskar. *Introduction to Statistical Relational Learning*, volume L of *Adaptive computation and machine learning*. MIT Press, 2007.
6. Neil Zhenqiang Gong, Ameet Talwalkar, and Lester Mackey. Jointly Predicting Links and Inferring Attributes using a Social-Attribute Network (SAN). *CoRR*, abs/1112.3, December 2011.
7. Pritam Gundecha and Huan Liu. Mining Social Media: A Brief Introduction. *Tutorials in Operations Research - New Directions in Informatics, Optimization, Logistics, and Production*, pages 1–17, 2012.
8. Prachi Joshi and Parag Kulkarni. Incremental Learning: Areas and Methods–A Survey. *International Journal of Data Mining & Knowledge Management Process*, 2(5), 2012.
9. Minkyoung Kim, David Newth, and Peter Christen. Trends of news diffusion in social media based on crowd phenomena. In *Proceedings of the Companion Publication of the 23rd International Conference on World Wide Web Companion*, WWW Companion '14, pages 753–758, Republic and Canton of Geneva, Switzerland, 2014. International World Wide Web Conferences Steering Committee.
10. WJ Li and DY Yeung. Relation regularized matrix factorization. In *IJCAI-09*, IJCAI'09, pages 1126–1131, San Francisco, CA, USA, 2009. Morgan Kaufmann Publishers Inc.
11. Ulrike Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.
12. Sofus A. Macskassy and Foster Provost. A simple relational classifier. In *Proceedings of the Second Workshop on Multi-Relational Data Mining (MRDM-2003) at KDD-2003*, pages 64–76, 2003.
13. Lei Tang and Huan Liu. Leveraging social media networks for classification. *Data Min. Knowl. Discov.*, 23(3):447–478, November 2011.
14. Duc Kinh Le Tran, Cécile Bothorel, and Pascal Cheung-Mon-Chan. Incremental learning with latent factor models for attribute prediction in social-attribute networks. In *EGC*, pages 77–82, 2014.
15. Yunhong Zhou, Dennis Wilkinson, Robert Schreiber, and Rong Pan. Large-Scale Parallel Collaborative Filtering for the Netflix Prize. AAIM '08, pages 337–348, Berlin, Heidelberg, 2008. Springer-Verlag.