Situation-aware User Interest Mining on Mobile Handheld Devices

Doreen Cheng, Henry Song, Swarooop Kalasapur, Sangoh Jeong
Samsung Electronics R&D Center, 75 W. Plumeria Dr. San Jose, CA 95134
(doreen.c, henry.song, s.kalasapur, sangoh.j)@samsung.com

Abstract. We present our experience in creating a novel unsupervised clustering algorithm for situation-aware pattern extraction from usage logs. The algorithm automatically estimates near-optimal number of clusters and cluster centroids. It models situation by taking advantage of sensors. 5-fold cross validations using real-world data show that the algorithm delivers higher accuracy than existing algorithms with much lower complexity. As a result it is the first clustering algorithm that can be practically deployed on mobile handheld devices in the real world. We also describe the research problems in situation-aware personalization that must be addressed before users can benefit from the learning algorithms and speculate on possible approaches to the solutions.

Keywords: interest mining, situation-aware, unsupervised learning, clustering, pattern extraction

1. Introduction

Consumers are overwhelmed when trying to get what they want from the Web using their handheld devices. Personalization is recognized as a key to help the users, and understanding user interests is critical for personalization. Since users' interests are often situation dependent, we decided to pursue research on situation-aware user interest mining. With more sensors, such as GPS, camera, and gyroscope, embedded in mobile handhelds, we can capture the environmental contexts while recording a user usage. Increased usage of wearable sensors can enable us to capture users' physical and physiological state. If we use the captured sensor data to represent the situation of a usage event and extract patterns from the usage logs, we can then use the patterns to predict the user's situational interests for better personalization. The problem is that handheld devices have limited power and computation resources for complex analytics. To address this problem, the Maggitti project [16] suggested sending usage logs to a server for the analysis. This approach, however, resulted in immediate privacy concerns.

To protect user privacy, we aimed for complete client solutions on mobile handheld devices. Currently there are many client-side solutions. Many of them target at media recommendations [1, 2] and Web information retrieval [3]. Most of them require profiles and situations defined by the users or by rules specified at design time; some even require the users to proactively train the system first. These requirements are unrealistic, since users are unlikely to specify situational preferences and keep them updated. Users are even less likely to train the system by entering their situations every time they use their phones. To alleviate users from these burdens, the author of [5] used unsupervised clustering algorithms to extract patterns from user usage data combined with context data from sensors. However, there is no substantial performance results reported and it is not generally applicable in real-life cases since it requires prior knowledge of K, the number of clusters, an important parameter for good performance, but K is typically unknown and varies from dataset to dataset.

For a client-side solution to be practically deployable in the real world, we must eliminate all the impractical assumptions and requirements. In addition, the solution must be lightweight and able to perform well for the datasets with varying characteristics such as the density of valid data entries. In this paper, we will first describe our approach to solving the problem in Section 2. We will then present the results in Section 3. In Section 4, we will discuss the difficulties in situation-aware pattern mining and in Section 5 we will conclude and speculate future research directions in this research area.

2. Approach

Our goal is to create a learning algorithm for usage pattern extraction that is practically deployable on mobile handheld devices. We chose two use scenarios to drive algorithm development: the situation-aware mobile task recommendation [6] and smart search and

advertisement [17]. In addition to their apparent values to end users, these scenarios also require different partitioning of the fields of the input vectors and exhibit different data characteristics, which means that any algorithm we develop should deliver good performance over a broader range of data characteristics.

Figure 1 depicts our general approach. User's usage is logged. Each log record contains three parts: a context part, an application usage part, and a part that represents user's interests. The context part contains the sensor data sampled at the time of usage. The application usage part captures the application being used. The interest part captures user's explicit and implicit feedbacks such as user ratings and selections.

The raw log data is processed and encoded for clustering based on the requirements of the driving use scenarios. A data vector is semantically partitioned into two portions: a portion representing a situation and a portion representing user-need to be predicated. For example, for task recommendation, situation is represented by the context part only and the application usage part is used for predicting the applications that are likely to be used in a situation. In case of smart search, both the context and the application usage are used to represent situation and the interest part is used for predicting user needs in the situation.

At a prediction time, the sensor values are sampled to represent the current situation that is compared with the situation portion of each centroids. The centroid most similar to the current situation is selected for prediction. For example, in the case of application recommendation, we recommend applications that are most frequently used in a current situation. In the case of smart search, we use highly probable keywords plus current location to form queries and search for links and Ads for the user.

The input data densities of the two driving use cases are quite different. By data density, we mean the density of valid entries in the prediction portion of an input matrix. In the case of task recommender, it means the density of the valid entries in the application-usage portion of the matrix, where as in the smart search case, it means the density of the valid entries in the user-interest portion of the matrix. In the task recommender case, we assume the density is 100% since an application is either used or not used. In the smart search case, we assume that only the data entries that contain user implicit or explicit feedbacks are valid, and all the others are unknown – similar to the assumption used in collaborative filtering. Since a typical user would only give feedbacks to a very small number of items among all the items considered, the data density is usually very low.

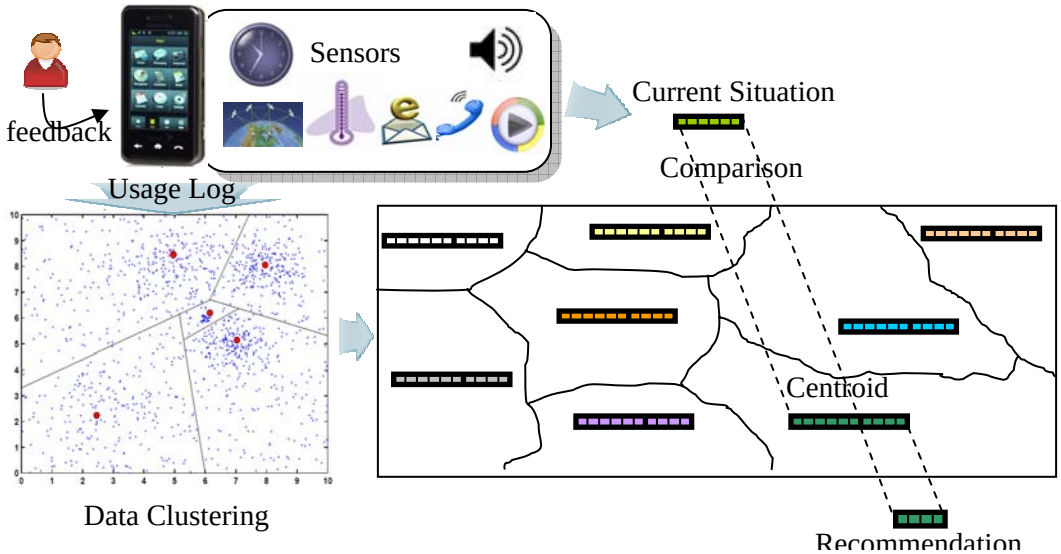

Fig 1. Extracting situational usage patterns for recommendation.

Seeking for an algorithm that could deliver good performance for both scenarios, we need datasets with various densities in our experiments. The first problem we faced is that there are no real life data that can be used for algorithm development and we had to devise a cold-start method to statistically generate synthetic data [6] that is as close as possible to real-life data. Meanwhile we conducted an 8-user, 6-month user study to collect real-world usage data for algorithm validation.

We chose to use unsupervised clustering approach to solving the problem because in the real life, mobile device usage data will not come with labels. We started by using the LBG [18] (Linde, Buzo, Gray) algorithm for its initialization stability and the co-clustering [8, 9] for its ability to capture coherent as well as homogeneous trends latent. We tried various ways of clustering the data, e.g. clustering the situation portion only, the prediction portion only, and both portions combined. We tuned the parameters of each algorithm to achieve good accuracy [19]. The problem is that all these algorithms require prior knowledge of *K*, the number of patterns contained in a given input dataset and a good initial guess of centroids for good accuracy. With gained insights of the problem, we set out to eliminate this requirement. There are algorithms for estimating optimal *K*, e.g. the *KL* algorithm [11], and algorithms for estimating optimal initial values of centroids, e.g. the KKZ algorithm [21]. But we are not aware of any algorithm can do both. In addition, all these algorithms are far too complex to be used on mobile handheld devices.

3. Results

We created the One-Pass Clustering (OPC) algorithm [20], a special-purpose algorithm for situation-aware usage pattern extraction. It is the first algorithm that can automatically make good estimates for both *K* and initial cluster centroids. It does so by taking advantage of situation awareness. It first classifies the input vectors according to the state combination of the situation-portion of each vector. It then classifies the groups into basic groups and noisy groups based on whether there are sufficient statistical evidences for prediction (i.e. enough valid data in the prediction-portion of a group). In other words, a basic group contains enough valid data to give high confidence for pattern-based prediction, whereas a noisy group does not. The OPC algorithm uses three thresholds for the classification. To qualify a group as a basic group, we first use a threshold to ensure that it has enough number of input vectors. We use a second threshold to ensure that the group has enough valid values in each column, a third threshold to ensure that that the centroid of the group has enough valid entries.

After all the groups are classified, the number of basic groups becomes the estimated optimal number of clusters, *K*, and the situation state of each basic group becomes the initial situation centroid for the cluster. The OPC algorithm then assigns the data instances in the noisy groups into appropriate clusters by computing the similarity between the situation states between a noisy group and each of the centroids and assigns the data instances in the noisy groups to the cluster with highest similarity. It then updates the corresponding situation centroid. After all the data in noisy groups are assigned to the clusters, the algorithm then re-computes the prediction portion in every cluster.

We compared the performance of the algorithms for both task recommendation and smart search using real-life data gathered from eight users including students, engineers, and managers for 6 months. We recorded their application usages on BlackJackII phones together with the date, time, location, battery level, device mode (e.g., “out door”, “automatic”, “vibrate”, etc.), and ring status at each usage event. .Limited by space, we will only include the co-clustering in this paper. We will also only describe the details of the result for the task recommendation here so as to motivate our discussions on the difficulties of situation-aware pattern extraction. Interested reader can find more details in [19][20].

We used a simple average method (AVG) as the base for algorithm comparison, where the unknown data fields are filled using the average of the valid values from the same column as predicated values; if there are no valid values in the column, the average of the valid values of entire matrix is used. For the co-clustering, we used the KL algorithm to estimate near-optimal number of clusters and used the LBG algorithm for initializing the centroids before running the

co-clustering itself. The KL algorithm iteratively finds the k that gives the smallest difference in the overall distortions between k^{th} and $(k+1)^{th}$ iterations. Its computation complexity is $O\{C^2mnT)\}$, where C is the number of clusters, T is the number of iterations, m and n are the dimensions of the situation portion of the matrix. Obviously the complexity is too high to be used on mobile handhelds, especially considering the combined complexity of co-clustering and KL. In comparison, the computation complexity of OPC, including k estimation and clustering, is $O\{Cmn\}$. Although theoretically the number of clusters could be equal to the number of state combinations of the situation portion of the vector, in reality, it is much smaller since the number of patterns in a dataset is typically small. In our 8-user-6-month datasets, the number of clusters ranges from 1 to 280.

		User 1	User 2	User 3	User 4	User 5	User 6	User 7	User 8
True Positiveness	CC	0.678	0.638	0.638	0.628	0.746	0.765	0.677	0.600
	OPC	0.815	0.699	0.892	0.715	0.880	0.865	0.762	0.653
	AVG	0.755	0.707	0.834	0.657	0.857	0.867	0.774	0.599
True Negativeness	CC	0.749	0.716	0.716	0.758	0.765	0.774	0.728	0.751
	OPC	0.767	0.725	0.752	0.773	0.784	0.790	0.743	0.760
	AVG	0.759	0.727	0.743	0.763	0.781	0.790	0.745	0.751
Overall Accuracy	CC	0.740	0.705	0.705	0.738	0.762	0.773	0.720	0.729
	OPC	0.773	0.722	0.770	0.764	0.796	0.800	0.746	0.745
	AVG	0.759	0.724	0.755	0.747	0.791	0.801	0.749	0.729

Table 1: Top-3 Recommendation Tests Results

We used true positiveness (TP), true negativeness (TN), and overall accuracy defined in [12] for measuring accuracy, since they are commonly used for testing accuracy of 2-class classifications and our algorithms make 2-class predictions as recommended or not. Since mobile handheld devices have limited screen size, we choose to recommend top-3 frequently applications. Table 1 shows the 5-fold cross-validation results.

From the table, we can observe that co-clustering performs the worst although it is the most complex algorithm especially combined with the LBG and KL algorithms. Although it might be possible to further tune the KL algorithm and recommendation schemes, we expect the improvements to be small and not worth the high computation and memory costs. On the other hand, the simple AVG algorithm performs well, especially for datasets that do not contain clear situational patterns or for datasets with no statistically sufficient data for pattern extraction.

We observed that the real-life usage data vary widely from having no clear patterns to having strong patterns. Also for a single user, typically there are no clear patterns in the usage data at the beginning and patterns emerge and become clearer over time. In addition, over time some old patterns will gradually disappear and new ones will emerge. This means that a practically deployable algorithm should also be able to deliver good accuracy in most of these cases in addition to being able to estimate K. Our results show that the OPC algorithm performs the best over the datasets with these varying characteristics. For example, in Table 1, the TP performance of OPC and AVG differs from 6% to -1%, while TN performance differs from 2% to 0%. Overall, the OPC performs better than AVG, because it behaves like the AVG algorithm when there are no sufficient evidences for patterns, but it behaves like clustering when evidences increase.

4. Discussion

We encountered problems in both data preprocessing and computation formulation. In this section we will present our experiences so as to invite discussions from the research community. The first problem is accuracy. As shown in the last section, even with good estimates of the number of clusters and initial centroids, the prediction accuracy is only moderately higher than the simple average method. This is mainly due to the difficulties in situation representation. For example, user's preferences often depend on her mood and who are around, but we are not able to capture these parameters. Using fixed context states may introduce noises since what context

state should be used for representing a situation may vary from user to user and from time to time. A naive solution could be to assign weights based on specific use scenarios, but such solution may not be adaptive to different users. Furthermore a same context may be more relevant for one situation than another. This means that the weighting for a context may not be applicable to all situations. Since the features chosen by traditional feature selections [15] are relevant to all clusters, these algorithms do not apply here. .Other factors such as imprecise similarity computation of situations may also contribute to lowering the performance, since we have hard time to justify the semantic meanings of using either the Euclidean or the cosine distance measures for our purposes.

The second problem is raw log data processing. For consumers to benefit from situation-aware user modeling, the raw log data needs to be automatically transformed in order to be used by the learning algorithms. The transformation includes cleaning, quantization/grouping, and encoding. The quality of transformation significantly impacts the performance and how to transform typically depends on the use scenarios and user's life style. A reasonable way might be to allow a use scenario to specify the requirements and the system automatically does the job. But we found several issues need to be addressed for this to happen.

One issue is the need to handle missing context data in raw log records. For example the location information may be missing because getting GPS data is very slow and the cell tower information is not available during phone calls. The battery status may be missing due to system limitations. We addressed this problem by sampling context values periodically in addition to sampling them at the time of use and by deriving a missing value from the valid value(s) in a time window around the usage event. But this interpolation can introduce significant noise into prediction and periodic sampling consumes power. Perhaps support from sensor and system designers to provide high quality context data can be a better solution to the problem.

Another issue is the difficulties of automatic cleaning, quantizing and encoding of raw data. For example many locations do not have statistically significant number of usage events and many applications occur infrequently in a log. We addressed this problem by retaining 9 frequently visited ZIP-code level locations, by combining the infrequently visited locations into a single group, and by grouping the applications by types, e.g. grouping SMS, MMS and IM as messaging. The issue gets more complicated if we consider that different use scenarios may require different treatments. Since there can be many types of data being logged, many applications available, and potentially a large number of items are to be predicated or recommended in the real world, how to model the log data fields in order to flexibly support the needs of various scenarios still need more research.

5. Conclusion and Future Direction

Situation-aware personalization is becoming more important in mobile and pervasive computing. Previous works in this area use assumptions that prevent them to be practically deployable in the real world. In this paper, we described the OPC algorithm that does not require the impractical assumptions and is able to perform well for datasets with broader characteristics. It delivers higher accuracy with much lower complexity than existing algorithms and runs completely on a mobile handheld device. However for end users to benefit from the results of user modeling, we need to address the following research issues: 1) a data model that supports automatic preprocessing (cleaning, quantization, and encoding) of usage data based on given requirements, 2) more accurate ways to approximate user's situations using sensors, 3) more semantically correct ways for computing similarity between context values. We also speculate that a client and server coordination solution could go a long way. Specifically a user's device could send predicted preferences to a server without revealing the user's situation and the server offers recommendations based on the user's preferences and the preferences from the people similar to the user.

References

1. S. Dornbush, K. Fisher, K. McKay, A. Prikhodko, Z.Segall, “XPOD – A Human Activity and Emotion Aware Mobile Music Player,” proc. 2nd Int’l Conf. Mobile Technology, Applications and Systems, 2005, pp. 1 – 6.
2. Z. Yu, X. Zhou, D. Zhang, C. Chin, X. Wang, J. Men, "Supporting Context-Aware Media Recommendations for Smart Phones," IEEE Pervasive Computing, July-September, 2006, vol. 5, no. 3, pp. 68-75.
3. Yau, S.S.; Huan Liu; Dazhi Huang; Yisheng Yao, "Situation-aware personalized information retrieval for mobile Internet," proc. 27th Int’l Computer Software and Applications Conference (COMPSAC 2003), vol., no., pp. 639-644.
4. C.S. Jensen, “Research Challenges in Location-enabled M-Services”, In Proc. of the 3rd International Conference on Mobile Data Management (MDM 2002), Jan. 22002
5. J.A. Flanagan, “Unsupervised Clustering of Context data and Learning User Requirements for a Mobile Device,” proc. 5th Int’l Conf. Modeling and Using Context, LNCS 3554, Springer, 2005, pp. 155 – 168.
6. D. Cheng, H. Song, H. Cho, S. Jeong, S. Kalasapur, A. Messer, “Mobile Situation-aware Task Recommendation Application,” Proc. Next Gen Mobile Apps & Tech (NGMAST-08), IEEE CS press, 2008, pp 228 – 233.
7. S.C. Madeira, A. L. Oliveira, "Biclustering Algorithms for Biological Data Analysis: A Survey," IEEE Transactions on Computational Biology and Bioinformatics, vol 1, issue 1, pp 24 – 45, 2004.
8. H. Cho, et.al. , “Minimum Sum-Squared Residue Co-clustering of Gene Expression Data,” Proceedings of the Fourth SIAM International Conference on Data Mining (SDM), pp 114 – 125, April 2004.
9. A. Banerjee, et.al., “A generalized maximum entropy approach to bregman co-clustering and matrix approximation,” Journal of Machine Learning Research, vol 8, pp 1919 – 1986, 2007.
10. T. Hastie, R. Tibshirani, and G. Walther, “Estimating the Number of Data Clusters via the Gap Statistic”, Journal of the Royal Statistical Society, B 63:411 – 423, 2001
11. W. J. Krzanowski, and Y. T. Lai, “A Criterion for Determining the Number of Groups in a Data Set using Sum of Squares Clustering”, Biometrics, 1985, 44:23 – 34
12. Confusion Matrix, http://www2.cs.uregina.ca/~dbd/cs831/notes/confusion_matrix/confusion_matrix.html
13. B. Sarwar, G. Karypis, J. Konstan, J. Reidl, “Item-based collaborative filtering recommendation algorithms,” In Proc. 10th int’l Conf. on World Wide Web (WWW’2001) ACM, New York, NY, pp. 285-295.
14. M. Balabnovic, and Y. Shoham, “Fab: Content-based Collaborative Recommendation”, Communications of the ACM, Vol. 40, Issue 3, pp 66 – 72, Mar. 1997
15. L. Buriano, M. Marchetti, F. Carmagnola, F. Cena, C. Gena and I. Torre, “The Role of Ontologies in Context-aware Recommender Systems”, In Proc. of the 7th International Conference on Mobile Data Management (MDM 2006), May, 2006
16. H.C. Peng, F. Long, and C. Ding, “Feature selection based on mutual information: criteria of max-dependency, max-relevance, and min-redundancy”, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 27, No. 8, pp.1226-1238, 2005
17. M. Roberts, N. Ducheneaut, B. Begole, K. Partridge, B. Price, V. Bellotti, A. Walendowski, and P. Rasmussen “Scalable Architecture for Context-Aware Activity-Detecting Mobile Recommendation Systems”, http://www.parc.com/research/publications/files/6319.pdf
18. H. Song, S. Kalasapur, S. Jeong, and D. Cheng, “SmartSearch: Situation-Aware Web Search on Mobile Devices”, IEEE CCNC, 2009.
19. Y. Linde, A. Buzo, and R. M. Gray, “An Algorithm for Vector Quantization Design,” IEEE Transactions on Communications, vol. 28, no. 1, pp. 84–95, January, 1980.
20. S. Jeong, D. Cheng, H. Song, and S. Kalasapur, "NON-COLLABORATIVE INTEREST MINING FOR PERSONAL DEVICES”, IEEE CIDM 2009

21. H. Song, S. Jeong, D. Cheng, and S. Kalasapur, "Efficient situation-aware user preferences mining on mobile devices", to be submitted to RecSys 2009.
22. I. Katsavounidis, C.-C. J. Kuo, and Z. Zhang, "A New Initialization Technique for Generalized Lloyd Iteration", IEEE Signal Processing Letters, Vol. 1, No. 10, Oct. 1994.