Proceedings of the IBM Austin Center for Advanced Studies 5th Annual Austin CAS Conference, Austin, TX, February 2004

Semisupervised Clustering for Intelligent User Management

Sugato Basu, Mikhail Bilenko, Raymond J. Mooney
Department of Computer Sciences
University of Texas at Austin
{*sugato, mbilenko, mooney*}*@cs.utexas.edu*

IBM Technical Contact: Mark W. Johnson
Tivoli Technical Strategy
IBM Software Group

Abstract

Grouping users automatically based on their system usage can be beneficial in an autonomic computing environment. Clustering algorithms can generate meaningful user groups that provide important insights to system administrators about user profiles and group policies. In particular, if a small amount of supervision is provided by the administrator to the clustering process, semi-supervised clustering algorithms can use this supervision to generate clusters which are more useful for user management. In this work, we demonstrate the utility of semi-supervised clustering in intelligent user management. We collect publicly available system usage data of users in a university computing environment, and cluster the users using semi-supervised hierarchical agglomerative clustering based on the profile of the processes they run. Initial supervision is provided in the form of a few users running a specific process. Semi-supervised clustering gives us more meaningful clusters than unsupervised clustering in this domain, demonstrating that our technique can find interesting and useful groups in data with minimal user intervention.

1 Introduction

The goal of autonomic computing is to help automate the management and maintenance of information technology in order to ease the burden on system administrators [15]. As IT systems become more complex and the number of users increases dramatically, automation of user management tasks becomes a high priority for system administrators. Automatic intelligent grouping of users based on their interactions with the system can serve two purposes. First, it can supply administrators with a global view of the user pool via meaningful user classes that are based on actual system usage information. Second, given expressive clusters of users, policies that apply to identified user groups can be developed along with profiles corresponding to particular clusters, alleviating the need to tailor group management tasks to a diverse set of users. Thus, grouping users based on their system usage provides insights about the set of users that can be directly translated into typical user profiles and policies for user groups.

Clustering algorithms are machine-learning and data-mining techniques that group data into meaningful categories containing similar objects [8, 9]. A typical data-mining application of clustering is grouping a company's customers into coherent groups based on the items that they have purchased. Clustering algorithms have been developed for a wide variety of data representations including vectors of features (with nominal or continuous values), strings, trees, graphs, etc.. By representing users with sets of features or statistics that characterize their system usage, data appropriate for clustering users can be developed. System-usage data provides a wealth of information about users, such as command traces, sets of running processes, and resource load profiles. Based on this information, meaningful user groups can be computed and proposed to system administrators. In our initial experiment on this idea, we have applied standard hierarchical agglomerative clustering (HAC) [14] to executing-process data for 624 users of our department's pool of public Linux workstations. Inspection of the resulting clusters revealed a number of meaningful groups, such as members of particular computer-science classes or research groups.

However, although the results of generic clustering algorithms will reveal regularities in the system-usage data, they may not represent a grouping that serves the purposes of a given system administrator. Clustering is called *unsupervised learning* since it does not use any user feedback to guide the classes it discovers. Typical *supervised learning* techniques, on the other hand, assume every training example is labeled with a desired class and the goal is to construct a categorization function appropriate for classifying future examples [18, 8, 9]. Recently, *semisupervised learning* techniques that exploit both labeled and unlabeled data have been developed [5, 12, 19]. In particular, *semisupervised clustering* employs a small number of examples for which the user has provided class labels or constraints, in order to bias the grouping so that it corresponds more closely to the user's view of similar and dissimilar objects.

We have developed several techniques for incorporating limited supervision into clustering, such as seeding (initializing clusters with the labeled objects) and distance-metric learning (training a similarity measure using the labeled objects) [2, 4, 3]. If a system administrator is willing to provide a few examples of users that they believe should (or should not) be in the same cluster, these algorithms can utilize such supervision to bias the clustering algorithm and help it produce results that are more useful for system management. An initial application of this idea to our Linux process data has demonstrated the ability of semisupervised algorithms to incorporate administrator preferences in user clustering.

2 Clustering

Clustering can be roughly defined as the problem of partitioning a dataset into disjoint groups so that objects belonging to the same cluster are similar, while objects belonging to different clusters are dissimilar. Traditionally, clustering has been viewed as a form of unsupervised learning, since no class labels on the data are provided. In the most general formulation, the number of clusters K is also considered to be an unknown parameter. Clustering problems can be categorized as generative or discriminative. In the generative clustering model, a parametric form of data generation is assumed, and the goal in the maximum likelihood formulation is to find the parameters that maximize the probability (likelihood) of generation of the data given the model. Clustering algorithms can also be categorized as either hierarchical or partitional [11], depending on whether the algorithm clusters the data into a hierarchical structure or gives a flat partitioning of the data.

In this work, we employ the Hierarchical Agglomerative Clustering (HAC) algorithm that constructs a hierarchy of clusterings in an bottom-up fashion [14]. In HAC, points are initially allocated to singleton clusters, and at each step the "closest" pair of clusters are merged, where closeness is defined according to a similarity measure between clusters. The algorithm generally terminates when the specified "convergence criterion" is reached, which in our case is when the number of current clusters becomes equal to the number of clusters desired by the user. Different cluster-level similarity measures are used to determine the closeness between clusters to be merged. Single-link cluster similarity considers similarity between clusters to be equal to similarity between two *most similar* points, leading to elongated clusters with good local coherence. Complete-link cluster similarity uses similarity of two most dissimilar cluster members which results in spherical clusters. Finally, group-average similarity takes the average similarity between all pairs of objects that lie in two clusters, striking a balance between single-link and complete-link clustering [17].

Different HAC schemes have been recently shown to have well-defined underlying generative models – single-link HAC corresponds to the probabilistic model of a mixture of branching random walks, complete-link HAC corresponds to uniform equal-radius hyperspheres, whereas group-average HAC corresponds to equal-variance configurations [13]. So, the HAC algorithms can be categorized as generative clustering algorithms.

3 Semi-supervised Clustering and Intelligent User Management

Semi-supervised clustering, where a small amount of initial supervision is used to aid and bias the clustering of the data, has been the focus of several recent projects [2, 16, 21, 22]. In clustering for intelligent user management, the initial supervision would come from information provided by system administrators. The supervision can include class labels on parts of the data, e.g., the administrator can label some users with a known set of processes mainly run by that user. It can also be provided in the form of constraints, e.g., the administrator can specify whether two users should be put in the same cluster, or whether they should not be in the same cluster. Semi-supervised clustering will be able to use the provided seeds or constraints to generate a clustering of the data that incorporates the preferences of the system administrator, thereby facilitating user management.

Existing methods for semi-supervised clustering fall into two general approaches that we call *search-based* and *similarity-based* methods. In search-based approaches, the clustering algorithm itself is modified so that user-provided labels or constraints are used to bias the search for an appropriate partitioning. This can be done by several methods, e.g., modifying the clustering objective function so that it includes a term for satisfying specified constraints [7], enforcing constraints to be satisfied during the cluster assignment in the clustering process [21], doing clustering using side-information from conditional distributions in an auxiliary space [20], and initializing clusters and inferring clustering constraints based on neighborhoods derived from labeled examples [2].

In similarity-based approaches, an existing clustering algorithm that uses a similarity metric is employed; however, the similarity metric is first trained to satisfy the labels or constraints in the supervised data. Several similarity metrics have been used for similarity-based semi-supervised clustering, including string-edit distance trained using EM [4], Jensen-Shannon divergence trained using gradient descent [6], Euclidean distance modified by a shortest-path algorithm [16], or Mahalanobis distances trained using convex optimization [10, 22]. Several clustering algorithms using trained similarity metrics have been employed for semi-supervised clustering, including single-link [4] and complete-link [16] agglomerative clustering, EM [6, 10], and KMeans [10, 22].

To simulate initial supervision provided by system administrators, we provided the clustering algorithm with process seed clusters, i.e., sets of users characterized by running specific processes, each seed cluster having the set of users that run a particular process. The clustering algorithm, on being given this initial supervision, was able to grow the seed clusters by adding users running similar processes, as shown by the initial results outlined in Section 4. Such a clustering could be useful to the administrator in a number of ways. For example, the administrator can run the clustering algorithm at various times and track the number of users in the cluster corresponding to a given process. The administrator can then determine the number of shared user-licenses that would be sufficient for the set of users.

Our results serve as a proof of concept, demonstrating that initial supervision in the form of important processes run by a subset of users was able to guide the clustering algorithm to get more useful user clusters than that obtained by purely unsupervised clustering. In a full semi-supervised user management system, data pertaining to user command traces and resource load profiles would also be collected along with the set of running processes. In that case, supervision can be provided by the system administrator in many other ways. For example, to cluster users based on their load profiles, the administrator could seed three clusters by heavy, medium and light users, and the resulting clustering would be biased towards partitioning the users based on their system usage. This information could then be used to automatically allocate more resources, e.g., network bandwidth, to heavy system users.

4 Data Collection and Initial Experiments

For our initial prototype we have attempted to obtain meaningful user groups from a pool of 624 users of public Unix workstations in the Department of Computer Sciences at the University of Texas at Austin. We have monitored processes on 186 workstations over a period of one month, taking hourly snapshots using the *ps* command and recording the processes logged by the snapshots into a central MySQL database. There was a total of 210 distinct processes logged. It is important to note that the vast majority of processes are of little value in distinguishing between groups of users, since many are sub-processes launched by window managers (e.g. *kdeinit* or *FvwmButtons*) or common utilities (e.g. *mozilla-bin* or *acroread*).

We converted the collected information into a dataset that associated each user u with a vector $\mathbf{x_u}$ of attributes. Each attribute corresponds to a process, and a non-zero value indicates that the user has run the process during the observation time. We have utilized the TF-IDF (term-frequency inverse-document-frequency) conversion frequently used in information retrieval to obtain meaningful feature weights x_{up} from raw usage data [1]:

$$x_{up} = \frac{freq(u,p)}{\max_{p'} freq(u,p')} \log \frac{N}{N(p)} \qquad (1)$$

where $freq(u,p)$ is the number of times user u has run process p; N is the total number of users; and $N(p)$ is the number of users that ran process p. The TF-IDF weighting scheme allows assigning more importance to processes that are run frequently by a given user while being uncommon across the entire user pool.

We first ran the HAC algorithm on the dataset of 624 210-dimensional vectors until similarity between clusters being merged was below a specified similarity threshold (0.8 in our experiments). The results were disappointing, as most users were clustered together because they ran common utility programs or window managers. Based on these results, we excluded 51 processes from the data description, leading to a more meaningful representation of system usage by the users. However, unsupervised clustering still yielded unsatisfactory results – the main reason for this is that the users could be clustered in multiple

user	**CART Group?**	**processes**
user1	Y	`aterm, rlogin, condor_shadow.s, eclipse, condor_shadow`
user2	Y	`condor_shadow, condor_shadow.s, oafd, gvim, xdvi.bin`
user3	Y	`condor_shadow.s, condor_shadow, 686, gdb, wish`
user4	Y	`condor_shadow.s, condor_shadow, gvim, sim-alpha, gdb`
user5	Y	`condor_shadow.s, condor_shadow, netkit-rlogin, gvim, rxvt, sim-alpha`
user6	Y	`condor_shadow.s, 686`
user7	N	`condor_shadow.s, sftp-server, netkit-rlogin, more`

Figure 1: A sample cluster containing users from the same research group

ways which were not necessarily informative; e.g. users that used the *VNC* remote access software were grouped together. While such clustering could be of interest if system administrators needed to group users by their location, it was of no use to us since we were interested in grouping users by their research interests or classes they were taking.

Next, we employed the semi-supervised version of HAC, where initial seeding was obtained by creating clusters of users that ran processes indicating their research interests, such as *hugs* (a Haskell language interpreter), *xspim* (a MIPS simulator), and *sim-alpha* (part of the toolset for the SimpleScalar architecture). The results improved considerably. For example, Figure 1 shows the cluster containing 6 members of the Computer Architecture and Technology Laboratory that was obtained based on seeding with two users who ran the *sim-alpha* process. Such initial seeding allowed bringing in more people using another process, *condor_shadow*, that many users in that group ran.

Other notable clusters that were obtained after seeding include undergraduate students who ran either *hugs* or *spim* processes. We conjecture that such clusters include users that are taking the same classes and hence utilize system resources in a similar manner.

5 Future Work and Conclusions

As demonstrated by our preliminary results on applying HAC to Unix process data, clustering algorithms from machine learning can be used to find interesting and meaningful groups in populations of users by analyzing statistics on their system usage. In particular, our results indicate that by supplying a very small amount of supervision, the discovered groups can be tailored to the specific goals of the system administrator. By combining the strengths of both supervised and unsupervised learning, semisupervised clustering can discover interesting and useful groups in data with only minimal human intervention.

Once meaningful groups of users have been automatically indentified, they can be used to intelligently allocate resources to maximize these users' productivity and satisfaction with the system. They can also be used to predict and anticipate new users' needs. Based on limited data from a new user's initial system utilization, he or she can potentially be identified as a member of a previously discovered cluster. Based on the known behavior of typical users in this group, their future needs can be reliably predicted and their system automatically configured with the appropriate applications and resources to support these demands.

More generally, IBM's vision of autonomic computing requires systems that are adaptive to their environment and that automatically detect regularities in the world and exploit them to maximize their performance and minimize the need for human intervention when there is a change. Research in machine learning and data mining has developed a rich set of tools for automatically discovering patterns in data and using them to make accurate predictions. Appropriate use of these tools will be extremely helpful in making the vision of autonomic computing a reality. Although our work on clustering users is only an initial step in this direction, we believe it is indicative of the important role than machine learning algorithms can play in making future computing systems more robust and adaptive.

References

[1] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press, New York, 1999.

[2] S. Basu, A. Banerjee, and R. J. Mooney. Semi-supervised clustering by seeding. In *Proceedings of 19th International Conference on Machine Learning (ICML-2002)*, pages 19–26, 2002.

[3] S. Basu, A. Banerjee, and R. J. Mooney. Active semi-supervision for pairwise constrained clus-

tering. In *To appear in the Proceedings of the 2004 SIAM International Conference on Data Mining (SDM-04)*, 2004.

[4] M. Bilenko and R. J. Mooney. Adaptive duplicate detection using learnable string similarity measures. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2003)*, pages 39–48, Washington, DC, Aug. 2003.

[5] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the 11th Annual Conference on Computational Learning Theory*, Madison, WI, 1998.

[6] D. Cohn, R. Caruana, and A. McCallum. Semi-supervised clustering with user feedback. Technical Report TR2003-1892, Cornell University, 2003.

[7] A. Demiriz, K. P. Bennett, and M. J. Embrechts. Semi-supervised clustering using genetic algorithms. In *ANNIE'99 (Artificial Neural Networks in Engineering)*, Nov. 1999.

[8] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, San Francisco, 2000.

[9] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer Verlag, New York, Aug. 2001.

[10] A. B. Hillel, T. Hertz, N. Shental, and D. Weinshall. Learning distance functions using equivalence relations. In *Proceedings of 20th International Conference on Machine Learning (ICML-2003)*, 2003.

[11] A. K. Jain, M. N. Myrthy, and P. J. Flynn. Data clustering: A survey. *ACM Computing Survey*, 31(3):264–323, 1999.

[12] T. Joachims. Transductive inference for text classification using support vector machines. In *Proceedings of the Sixteenth International Conference on Machine Learning (ICML-99)*, Bled, Slovenia, June 1999.

[13] S. D. Kamvar, D. Klein, and C. D. Manning. Interpreting and extending classical agglomerative clustering algorithms using a model-based approach. In *Proceedings of 19th International Conference on Machine Learning (ICML-2002)*, 2002.

[14] L. Kaufman and P. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley and Sons, New York, 1990.

[15] J. O. Kephart and D. M. Chess. The vision of autonomic computing. *IEEE Computer*, 36(1):41–50, 2003.

[16] D. Klein, S. D. Kamvar, and C. Manning. From instance-level constraints to space-level constraints: Making the most of prior knowledge in data clustering. In *Proceedings of the The Nineteenth International Conference on Machine Learning (ICML-2002)*, Sydney, Australia, 2002.

[17] C. D. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA, 1999.

[18] T. Mitchell. *Machine Learning*. McGraw-Hill, New York, NY, 1997.

[19] K. Nigam, A. K. McCallum, S. Thrun, and T. Mitchell. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39:103–134, 2000.

[20] J. Sinkkonen and S. Kaski. Semisupervised clustering based on conditional distributions in an auxiliary space. Technical Report A60, Helsinki University of Technology, 2000.

[21] K. Wagstaff, C. Cardie, S. Rogers, and S. Schroedl. Constrained K-Means clustering with background knowledge. In *Proceedings of 18th International Conference on Machine Learning (ICML-2001)*, 2001.

[22] E. P. Xing, A. Y. Ng, M. I. Jordan, and S. Russell. Distance metric learning, with application to clustering with side-information. In *Advances in Neural Information Processing Systems 15*. MIT Press, 2003.