Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI-10)

Assisting Users with Clustering Tasks by Combining Metric Learning and Classification

Sumit Basu
Microsoft Research
Redmond, WA
sumitb@microsoft.com

Danyel Fisher
Microsoft Research
Redmond, WA
danyelf@microsoft.com

Steven M. Drucker
Microsoft Research
Redmond, WA
sdrucker@microsoft.com

Hao Lu
University of Washington
Seattle, WA
hlv@cs.washington.edu

Abstract

Interactive clustering refers to situations in which a human labeler is willing to assist a learning algorithm in automatically clustering items. We present a related but somewhat different task, *assisted clustering*, in which a user creates explicit groups of items from a large set and wants suggestions on what items to add to each group. While the traditional approach to interactive clustering has been to use metric learning to induce a distance metric, our situation seems equally amenable to classification. Using clusterings of documents from human subjects, we found that one or the other method proved to be superior for a given cluster, but not uniformly so. We thus developed a hybrid mechanism for combining the metric learner and the classifier. We present results from a large number of trials based on human clusterings, in which we show that our combination scheme matches and often exceeds the performance of a method which exclusively uses either type of learner.

Introduction

The daily lives of information workers abound with clustering and categorization problems. For example, a researcher must take hundreds of papers and book chapters from her field and organize them into coherent lectures; a program committee must take hundreds of accepted papers and group them into sessions and tracks; an admissions panel must take thousands of applicants and group them into appropriate departments. In many cases, there are underlying rules and metrics which explain much of how users are grouping the items; however, these metrics may not be obvious to them or easy for them to articulate. Furthermore, not all items will follow these metrics; there will be important outliers that cannot be missed. On the other hand, every group they create and every item they put into a group is evidence for how the other items should be grouped. The question, then, is whether and how a learning system might assist users with problems of this kind.

There has been a steady stream of work in the literature concerning “interactive clustering,” which initially seemed appropriate for this task. These methods take input from users in the form of “must-link” and “cannot-link” constraints, which specify whether two items belong together or apart. These constraints are then used to learn a distance metric, after which traditional clustering mechanisms such as k-means can be used to group items (more details are in our related work section). However, it was not clear to us that specifying such constraints was a natural part of users’ behavior for such tasks. We thus embarked on an observational study, in which we found that instead of specifying must-link and cannot-link constraints, users preferred to make semantically meaningful clusters and incrementally add items to them, i.e., they specified “must-belong” (and implicitly, “cannot-belong”) constraints between items and clusters.

While such labels differ in intent from those specified in interactive clustering, the mathematical framework of metric learning can easily be adjusted to incorporate these changes, as we will show in a later section. The user experience is quite different, though: the user is now asking for recommendations for a cluster given a set of items, or asking for a label given a new item. Both of these problems seemed amenable to a classification approach. Also, as this is a different problem setup than interactive clustering, we refer to this new scenario as *assisted clustering*.

To further explore how to help users with such tasks, we built a preliminary assisted clustering system and used it to collect ground truth data. We then experimented with both metric learning and classification approaches, and found that neither approach was always the winner for all clusters: classification would do better on some; metric learning would do better on others.

This led us to consider how we might combine the two approaches so that we could get the best of both worlds. While there are a variety of methods for combining sets of classifiers or rankers, our situation proved to be unique, as we will explain in our related work section. We thus developed a hybrid mechanism for doing this which converts the classifier for each cluster into a kernel for the metric learner. As we show in our results, this method matches and sometimes substantially exceeds the performance of a system that exclusively uses either the classifier or the metric learner approach.

In the remainder of this paper, we give greater detail to all of these steps: the related work, including interactive clustering and other means of combining algorithms; our own investigations, methods, and experimental setup; and finally, the results we found in our experiments and a discussion of their implications.

Related Work

The most relevant algorithmic work to our own is that of interactive clustering, such as the canonical papers by (Cohn and Caruana 2000) and (Bilenko, Basu, and Mooney 2004). As discussed above, these assume a set of user labels of items that must belong together and items that cannot be together (must-link and cannot-link constraints). Typically, these approaches make use of a metric learning approach, in which a distance function between items in learned. A common formulation is to propose a distance function between items $d(x_i, x_j)$ which is parameterized by a set of weights α_k over individual distance measures, often feature differences; the weights are then optimized to minimize a cost function that prefers must-link pairs to have small distances and cannot-link pairs to have large distances. An alternate approach is to optimize a transformation of the feature space f by a linear transform K such that appropriate Euclidean distances in the transformed space are small/large as implied by the constraints. In either case, given the new distance metric, traditional clustering methods (e.g., k-means) are used to automatically cluster all of the items. In other words, the user provides a set of labels, and the system produces a clustering. There has been very limited work on attempting to use these methods in an interactive *system*, in which users incrementally add items to the clusters; the principal work we know of is that of (Desjardin, MacGlashan, and Ferraioli 2007), though we note that it studies a simulation of user behavior and not an interactive system *per se*. In that work, the authors use the interactive clustering formulation of (Bilenko, Basu, and Mooney 2004) at each iteration. There is also some recent work on an interactive system for classification (Seifert and Lex 2009), but it does not address the clustering/categorization task.

As we described above, our formulation is somewhat different: instead of providing constraints between items, the users are placing items into meaningful clusters. Furthermore, the goal of our system is not to automatically cluster the items based on labels, but to provide appropriate recommendations for each clusters. Certainly metric learning is applicable here, as we can suggest for each cluster those items that are closest via the learned metric. However, since we also effectively have labels for which items belong to which cluster, we can also frame this as a classification problem. As we sought to combine the benefits of both of these approaches, we surveyed existing work on combining learners. The best known of such work is that on combining classifiers – there is a rich history of techniques such as boosting and bagging, as covered by (Bauer and Kohavi 1999), to achieve better results than a single classifier. However, a metric learner is not a classifier, and does not fit well into this framework. Similarly, there are means of combining ranking metrics such as RankBoost (Freund et al. 2003), but these do not give us a means of integrating classifiers. Furthermore, we have the unique advantage of an incremental context for each individual cluster, in which a combined learner can take advantage of labels in previous rounds to perform better in future rounds.

Finally, we note that classifier combination methods could certainly be applied to improve our base classifier's performance; similarly, ranking combination methods could be used to improve the metric learner. We are not claiming that our implementation of either of these components is either novel or optimal; our contribution is instead in (1) formulating the problem based on users' observed behavior, by which the applicability of both classifiers and metric learners become clear, and (2) proposing and evaluating a means of combining these two types of learners. We certainly expect that the overall performance of the system can be improved by replacing the individual learners with more sophisticated counterparts, and expect to pursue this in future work.

Investigating Sorting Behavior

To better understand how human subjects sort items in the real world, we set up an observational study in which five subjects were given one hour to manually sort sixty printed papers from the CSCW conference. All subjects were familiar with the area. We recorded video from several points of view as well as audio so we could observe the users' sorting behavior in detail. An analysis of this data helped us identify several key trends. First, subjects tended to create categories that were semantically meaningful to them based on an initial paper or two and then add papers to these; this differs from the model of creating must-link and cannot-link constraints assumed by interactive clustering methods. Second, subjects would often remember having seen a paper relevant to a current category that they had earlier set aside, and would then spend time hunting for it. Third, as they developed a set of categories, they tended towards sorting new items into the existing piles rather than creating new categories. Finally, when a paper from the pile was not relevant to the current set of categories, they would either put it somewhere on the desktop or add it to a "miscellaneous" pile.

After sorting the papers, we interviewed each subject to see what kinds of machine assistance they would have found most helpful. Most of the subjects desired search functionality, which would let them find documents they had seen but subsequently misplaced. Next, many asked for suggestions of new items for a cluster, especially once they had established firm categories. Finally, some users wanted suggestions for which category a new item from the pile should go into, though most felt that this was a much easier task than finding additional relevant items.

Interactive System and Data Collection

With our observations and the subjects' feedback in mind, we developed an interactive system for assisted clustering. The interface is a large virtual canvas with stacks of "baseball card" representations of data items; in the case of documents, the face of the card shows the title and authors. By double-clicking on the item, users can see more detail; for documents, this is the full abstract. A screenshot of the system is shown in Figure 1 below.

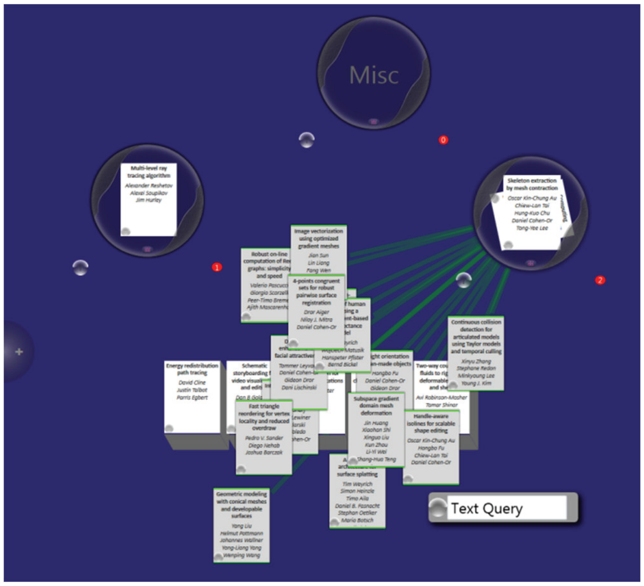

Figure 1. A screenshot of our assisted clustering system, showing suggested items for the selected cluster. Note that these recommendations were turned off while subjects were clustering items to provide ground-truth data for our experiments.

The core functionality of our system mirrors what we observed our subjects doing in the study: it allows for users to create new clusters, move items into those clusters, open up clusters to examine their contents, and move clusters around; we also provide a "miscellaneous" cluster. Beyond this, we added a variety of features to assist users with the task. First, we provide a search box which can retrieve items by word-based queries. Next, when a user clicks on a cluster, the best matching items from the unsorted pile are lifted from the pile, indicated via connecting lines, and moved towards the cluster (see Figure 1); at this point, the user can drag relevant items into the cluster. When the user releases the cluster, the untouched items snap back into the pile. Similarly, when a user clicks on an item, the best matching clusters are indicated by visual lines. Determining *which* items are the best matches, of course, is the subject of this paper, and we give more detail on this in the sections below.

In order to both validate our design and to collect labels for our investigations, we had four subjects use this system to sort sets of 300 documents from one of three academic communities (CSCW, SIGGRAPH, and SIGIR) into clusters. They were instructed to group as many items as possible, but were not required to put every item into a cluster. To prevent biasing from our algorithms, the recommendations of relevant items/clusters were not active; the subjects could only use search to help them with their manual sorting. We used the groupings the users came up with as ground truth labels for our experiments in the sections below.

Finally, we note that while the experiments in this paper all involve sorting documents, the system was developed to work for any type of data that can be represented visually and for which we can express a set of features and similarity metrics. We have in fact used the system for a different task, clustering structured feedback for software products, and will report on those studies in a future paper.

Retrieval with Metric Learners vs. Classifiers

In this section, we go through the details of our formulation for both the metric learner and the classifier and how we can use each one to find the best matching items in the assisted clustering scenario.

Metric Learning Approach

Given that we are seeking the "nearest" items in the appropriate metric space, it is quite natural to use metric learning to train a distance measure. The formulation we use is similar to existing work such as (Bilenko, Basu, and Mooney 2004), though we use must-belong and cannot-belong constraints vs. must-link and cannot-link. Consider a series of candidate distance measures, $d_k(x_i, x_j)$, which we wish to combine with weights α_k. We can then write the following combined distance function:

$$D(x_i, x_j) = \sum_k \alpha_k d_k(x_i, x_j)$$

In addition to the distance function, we have labels from the user as to which items must belong to which clusters; this also tells us that those items cannot belong to the other clusters. We can express the distance between the item and a cluster as the distance from the item to the centroid of the cluster, c_j, which is the item which has the minimum cumulative distance to all other items in the cluster. We want to minimize the distance between items and the clusters they belong to, and maximize the distance to other clusters. This leads to the following cost function which we wish to minimize:

$$C = \frac{1}{I_m} \sum_{\substack{i \in must \\ belong}}^{I_m} D(x_i, c_j) - \frac{1}{I_c} \sum_{\substack{i \in cannot \\ belong}}^{I_c} D(x_i, c_j) + \gamma \sum_k \alpha_k^2$$

Note that we normalize each type of constraint by the number of such constraints (I_m and I_c), as there are typically many more cannot-belong constraints. Also note that the final element is a regularization term, which controls the growth of the α_k; for our experiments γ was set to 1. To minimize this cost function C, we take the derivative with respect to the weights:

$$\frac{dC}{d\alpha_k} = \frac{1}{I_m}\sum_{\substack{i\in must\\ belong}}^{I_m} d_k\left(x_i, c_j\right) - \frac{1}{I_c}\sum_{\substack{i\in cant\\ belong}}^{I_c} d_k\left(x_i, c_j\right) + 2\gamma\alpha_k$$

We further constrain the α_k to be positive by representing them as β_k^2; we then do our optimizations in terms of the β variables via the chain rule:

$$\frac{dC}{d\beta_k} = \frac{dC}{d\alpha_k}\frac{d\alpha_k}{d\beta_k} = \frac{dC}{d\alpha_k}2\beta_k$$

Given these derivatives and the convexity of the formulation, we use the L-BFGS method (Nocedal 1980) to find the optimal β_k, which we square to find the α_k for the overall $D(x_i, x_j)$. Note that this optimization must be performed every time an item is added or removed from any cluster. We can then use this distance function to find the closest items to a cluster.

In order to apply the method, we also need to define some component distances $d_k\left(x_i, x_j\right)$. For our problem, we first converted the title, author, and abstract of each document into their vector TFIDF representations, a common approach for information retrieval tasks (Salton and McGill 1983). We then compute six distance measures: the ℓ_1 and ℓ_2 vector norms between the TFIDF representations of the titles, the authors, and the abstracts of documents x_i and x_j. In earlier experiments, we also added 20 random distance metrics; as expected, the weights for these quickly converged to zero in two or three examples.

Classification Approach

Since the user is placing items into distinct categories, we have positive and negative labels for each cluster, and can train a classifier for each cluster that determines which items from the unsorted pile may belong to it. Furthermore, for many classifiers, we can compute not only a binary answer but also a score value for how well the item fits. We chose to use logistic regression (Bishop 2006) both because of its interpretability, as it learns a weight for each feature f_j, and also for its meaningful output score, i.e., the probability that the given example's label y_i is 1:

$$f_c(x_i) = P(y_i = 1) = \frac{1}{1 + e^{-\Sigma w_j f_{ij} + \delta}}$$

We find the parameters w_j and the bias δ by minimizing the total log likelihood of the labeled data under this model, weighting the positive and negative examples so they have equal importance to the cost function; we also add an ℓ_1-regularization term to encourage sparsity in the solution (Ng 2004). As the formulation is convex, we again find the optimal solution using L-BFGS.

In this case, instead of distance measures, we need to supply a set of features to the learner. We begin with the TFIDF vector representation of the combined title, author list, and abstract, which is of 3502 dimensions and quite sparse. This results in having to learn 3502 parameters w_i, which unsurprisingly led to very poor performance in our initial experiments, even with the ℓ_1 regularizer. We thus reduce the dimensionality using PCA and project the raw vector onto the top 100 eigenvectors (i.e., corresponding to the 100 highest eigenvalues); this is typically referred to as latent semantic indexing or an LSI representation in the information retrieval literature (Berry, Dumais, and O'Brien 1995). Finally, once we have trained the classifier, we can produce a set of recommendations from this method by computing $f(x_i)$ for each uncategorized item and then sorting by the score value.

Finally, we note that there are many possible variations to both learners: for the metric learner, for instance, instead of the distance to the centroid of the cluster, we could use the minimum, maximum, or mean distance; instead of a single global distance function we could have a separate set of weights; we could also have chosen from a host of other possible component distance functions. For the classifier, there are many other choices of models and features we could have used which could certainly improve the performance. However, it is the combination of the two types of methods that is key to our work, as we detail in the next section.

Combining the Learners

Before we discuss how to combine the learners, let us consider why this might be a good idea. The two methods are learning quite different things: the metric learner is learning a global distance function between items, while the classifier is seeking a discriminating surface between items inside and outside a given cluster. Furthermore, when the user adds a new item to a cluster, it provides the same benefit to all clusters for the metric learner, but helps the particular classifier for that cluster more than the others. In a similar vein, if 50 items have been put into clusters when we create a new cluster with a single item, the metric learner will have likely converged to a set of weights, while the classifier must discriminate based on only one positive example. It is reasonable, then, to expect that one or the other method might be better in different situations. In fact, in our data, when comparing only these two methods, we found that the metric learner strictly beat the classifier in 23% of the trials, while the classifier did better 50% of the time (the rest were ties). Clearly it would be to our benefit if we could always do as well as best performing algorithm, or ideally even better.

As we discussed earlier, though, the prior work in combining classifiers or metrics does not apply well to our situation. As a result, we have developed a new hybrid mechanism by which we can combine the recommendations from each approach. We describe the method below, along with other metrics that will help demonstrate its relative performance.

Hybrid Metric Learner. While we cannot turn the metric learner into a classifier without assigning a threshold to its recommendations, we can turn the results of an individual classifier into a distance function that will only affect one cluster by careful assignment of its values. Specifically, for cluster c:

$$d_c(x_i, x_j) = \begin{cases} s(1 - f(x_i)), & x_j \ in \ c \\ 0, & otherwise \end{cases}$$

In other words, if the classifier thinks the item is in c, i.e., $f(x_i)$ is close to 1, the distance is small; otherwise it is large; if c is not involved, it contributes nothing. This unusual, asymmetric formulation has the advantage that only the columns corresponding to a particular cluster (and thus a particular classifier) are affected. As a result, though the weight for d_c will be learned globally for all items, it will only affect cluster c. The scaling function s accounts for the fact that the outputs of logistic regression hover around 0.5 when the classifier has not seen many examples, as is typical in our trials. Its form is as follows:

$$s(z) = \zeta(\ z - 0.5) + C$$

The values for C and ζ are set after all elements of d_c are computed, and normalize the values of d_c to be between 0 and 10 to be commensurate with the other kernels.

Best-Individual. This is the maximum of the performance of the individual methods, i.e., the classifier and metric learning schemes. As such, this is not an implementable scheme, but represents an upper bound of performance if we were able to omnisciently choose which individual method would work best for the given context.

Random. This is a baseline measure that selects items at random from the unsorted pile.

Experiments and Results

Our four subjects' manual sorting of the three corpora (see Section "Interactive System and Data Collection") resulted in 104 total clusters; each cluster contained between 1 and 32 members. We chose to perform our experiments on clusters of larger than 10 elements so that we could meaningfully compute learning curves for the methods (i.e., performance vs. the number of presented examples); this left us with 46 clusters.

Rather than only using the order in which the subjects placed items in clusters, we generated 10 times more trials by randomizing the order in which positive examples and negative examples were chosen. The positive examples were simply reordered; the negative examples were chosen via the "Chinese Restaurant Process" (Aldous 1983). In this process, an item is added to an existing cluster with probability α (0.8 in our experiments), where the particular cluster to receive a new item is chosen with probability proportional to the cluster's size. A new cluster is formed by choosing from the remaining items with probability $1 - \alpha$. This is a standard statistical model for cluster growth and ensures that clusters grow organically, with a few items being added to each, as opposed to many clusters appearing with a single example. Furthermore, this matched well with the observed behavior of our subjects.

This randomization resulted in a total of 460 unique runs over individual clusters (46 clusters times 10 re-orderings). To compute our results, we chose a target cluster and re-ordering, then added a pair of examples at a time, one positive and one negative, from the chosen ordering. At each step, we computed the performance for all the methods above. This resulted in 7250 unique (cluster, randomization order, number of examples) tuples. To compute the performance the methods for each tuple, we allowed each method to produce 10 suggested items for the target cluster given the examples thus far, and then tested what fraction of those suggestions were correct. This is referred to as precision-at-N (with N=10 in our case). Note that if the number of remaining items in the cluster was less than 10 for a given state, the fraction was in terms of the number remaining.

We also wished to investigate the difference in performance between a "cold start," in which the canvas would start with a clean slate (no clusters), after which positive and negative examples would be added as described, versus a "warm start," in which 50 items were already placed into other clusters (via the restaurant process) before we began working on the target cluster. The latter scenario represents the situation where the user has already sorted some data and has now started on a new cluster; this is of course the more common scenario in the interactive task.

We now show the results from these experiments in terms of both overall precision and the learning curves (per-label improvements).

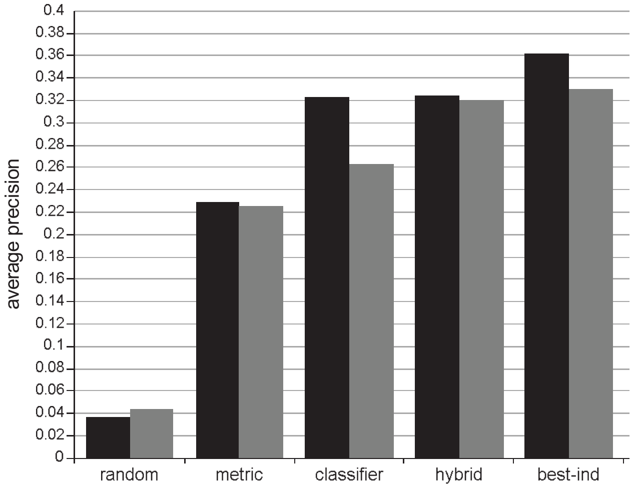

Figure 2: Average precision over all trials for all methods for cold start (black), i.e., no other clusters formed yet, and warm start (grey), i.e., 50 items already in other clusters. "Hybrid" is our method; "metric" and "classifier" refer to the individual learners; "best-ind" represents an unimplementable upper bound on the individual methods. See Section "Combining the Learners" for more details about each method.

Overall Precision. Figure 2 shows the average precision of each method over all 7250 data instances, i.e., over all combinations of clusters, randomizations, and numbers of positive/negative examples, under both cold and warm start conditions.

In all conditions, the hybrid method does at least as well as each individual method. In the cold start case, it does only incrementally better than the classifier (not statistically significant), but in the more typical warm start case, it beats both methods by a large margin. Note that all differences in performance are statistically significant at the p=0.001 level except for the case mentioned above (hybrid vs. classifier in the cold start condition).

It is interesting to note that the classifier performance decreases substantially in the warm start condition compared to the cold start; this is because the classifier is now faced with learning a much more complex decision boundary as specified by the many negative examples already in other clusters. A more complex classifier may have fared better in this case, but would in turn overfit the data and fare worse when the amount of data was small (i.e., the cold start condition). The advantage of our hybrid method is that it can leverage the strengths of both individual methods where appropriate: were we to incorporate the more complex classifier, we expect that our method would reduce the weights corresponding to the classifier kernels when little data was available, and rely on them more as more examples were added.

Performance vs. Number of Examples. For the next set of experiments, we wished to see the learning curves, i.e., how the performance changed with the amount of data presented to the various methods. Figures 3 (cold start) and 4 (warm start) show the results of these experiments. In both cases, the hybrid learner exceeds the performance of either component learner. As before, in the cold start case, it does incrementally better than the classifier and significantly beats the metric learner. In the warm start case, it substantially outperforms both methods and at times even exceeds the performance of the omnisciently chosen best-individual method. Note that this is possible as the hybrid method can incorporate information from both schemes.

As we discussed in our description of the relative strengths of classifiers vs. metric learners, we expected that the metric learner would benefit more than the classifier from the warm start. What we saw was that the metric learner did in fact see a slight improvement, but that the classifier saw a large drop in performance with respect to the cold start case, again presumably due to the larger number of available negative examples (all the warm start examples plus the ones paired with each new positive example). Though our classifier formulation balances the cost of positive and negative examples, the variety of negative examples clearly causes it problems, and the metric learner performs better for the first few examples.

It is possible that we could improve the classifier's performance by reducing the number of negative examples, but such an approach would require careful tuning to determine when and by how much to reduce the data; it is likely that data reduction would help in some cases and hurt in others. In contrast, our method is able to automatically adjust to the best combination of the component classifiers and distance metrics. The key observation from these results is that regardless of the individual methods' performances, the combined learner is able to consistently exceed them both.

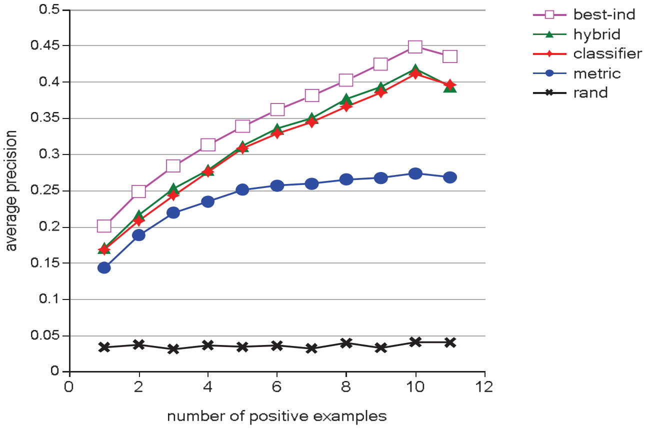

Figure 3: Average precision vs. number of positive examples from a cold start.

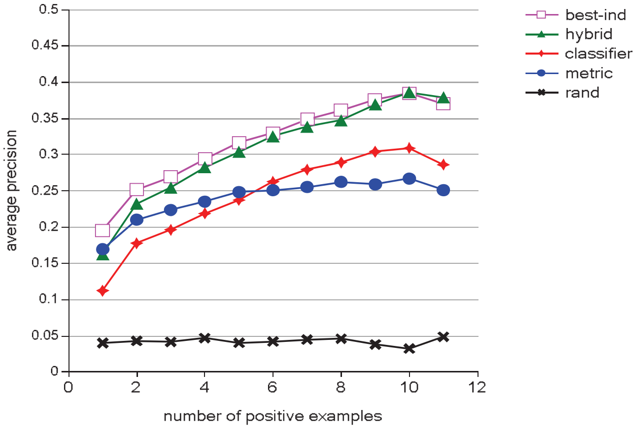

Figure 4: Average precision vs. number of positive examples where 50 items were already placed into other clusters (warm start condition).

Discussion

We have presented a novel problem space, assisted clustering, based on our observations of real users manually clustering items into categories. In this space, both classifiers and metric learners are appropriate models. In order to leverage the power of both, we have developed a means by which we can combine these disparate types of learners; we have shown that the resulting hybrid method performs as well or better than the individual learners. Furthermore, we are optimistic that this hybrid method may also be applicable in other scenarios where both types of learners are relevant.

References

Aldous, D. 1983. "Exchangeability and Related Topics", in *l'École d'été de probabilités* de *Saint-Flour, XIII–1983*, Berlin: Springer. pp 1–198.

Bauer, E. and Kohavi, R. 1999. An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants. *Machine Learning* 36(1-2): 106-139.

Berry, M. W., Dumais, S.T., and O'Brien, G.W. 1995. Using Linear Algebra for Intelligent Information Retrieval. *SIAM Review* 37(4): 573-595.

Bilenko, M., Basu, S., and Mooney, R.J. 2004. Integrating Constraints and Metric Learning in Semi-Supervised Clustering. In *Proceedings of the Int'l Conf. on Machine Learning (ICML).*

Bishop, C. M.. 2006. *Pattern Recognition and Machine Learning*. New York: Springer.

Cohn, D., and Caruana, R. 2000. Semi-Supervised Clustering: Incorporating User Feedback to Improve Cluster Utility. In *Proceedings of the Conf. on Artificial Intelligence*. AAAI Press.

Desjardins, M., MacGlashan, J., and Ferraioli, J. 2007. Interactive Visual Clustering. In *Proceedings of the Int'l Conf. on Intelligent User Interfaces*. ACM Press.Engelmore, R., and Morgan, A. eds. 1986. *Blackboard Systems.* Reading, Mass.: Addison-Wesley.

Freund, Y., Iyer, R., Schapire, R.E., and Singer, Y. 2003. An Efficient Boosting Algorithm for Combining Preferences. *Journal of Machine Learning Research* 4: 933-969.

Ng, A. 2004. Feature Selection, L1 vs. L2 Regularization, and Rotational Invariance. In *Proceedings of the Int'l. Conf. on Machine Learning (ICML).*

Nocedal, J. "Updating Quasi-Newton Matrices with Limited Storage." *Mathematics of Computation* 35: 773-782.

Salton, G. and McGill, M.J. 1983. *Introduction to Modern Information Retrieval*. New York: McGraw-Hill.

Seifert, C. and Lex, E. 2009. A Novel Visualization Approach for Data-Mining-Related Classification. In *Proceedings of Information Visualization*.