Programmer's Active Learning: A Broader Perspective of Choices for Real-World Classification Tasks that Matter (Position Paper)

George Forman
HP Labs, Palo Alto, CA, USA

Abstract. This position paper opens the discussion about a future kind of active learning where—rather than just asking a domain expert to assign class labels to items—the system directs a proficient data mining *programmer* to perform a much wider variety of tasks, e.g. writing code to produce more predictive features for distinguishing confused classes, composing regular expressions to extract key-value features from technical text, writing a classification rule for some tight cluster of cases found by the system, or deciding whether the current classifier is satisficing, in view of its limited rate of improvement. Since data mining programmers are already involved in most efforts to develop classifiers for important real-world tasks, the benefits of channeling their talents to optimize their productivity are intriguing, as well as the potential for reducing the time-to-market for deploying an accurate classifier.

Keywords: dual active learning, machine learning classification, interactive data mining, mixed-initiative user-interfaces, programmer productivity aids

1 Background

Active learning methods iteratively choose the next sample case(s) to be labeled by the domain expert from a stream or pool of unlabeled cases. Published papers often show a $\sim$10$\times$ reduction in the labeling effort to achieve a given (often mediocre) level of classification accuracy vs. random sampling.[1]

Recently there have been some works that deviate from *only* asking to label unlabeled cases. *Dual active learning* gives the system the additional option to ask the expert to label a particular *feature* as being indicative of one class, if appropriate (e.g. [17, 4, 11, 1]). For example, after many normal active learning rounds, the system could ask the expert to confirm or deny an increasing belief that the phrase 'reverses aging' should be associated with the SPAM class. Additionally, there has been some work that gives the active learning method the option to ask for re-confirmation or re-labeling of a suspiciously labeled training case (e.g. [16, 19, 18]).

[1] For surveys on active learning and variants, see [13–15, 9, 10].

These systems and others share the limited theme of asking a domain expert to look at something and select a simple multiple-choice among the classes (possibly with an option to balk). Such user-interfaces are fairly easy to build and require little skill to operate. Perhaps most importantly for the active learning research community, they are trivial to simulate in the many active learning studies and occasional active learning competitions [8, 3]: the testing harness simply reveals another hidden ground-truth label from the benchmark dataset. But when you—as a technically skilled data miner—are faced with a novel, 'real-world'[2] classification task in which you really need to maximize the classification accuracy, there are certainly many more options available, such as writing additional code for extracting more predictive features. Such options can be far more effective than just incrementally labeling more cases for the training set.

2 Vision and Challenge

Although many actions may be available to the motivated data mining expert programmer, the scope of their potential action space is dauntingly large, and it is nigh impossible to optimize one's choice of actions. In view of this difficulty and opportunity, we propose a new, broad-ranging idea: *advanced active learning systems to help guide the technically capable data miner to better optimize their choice of actions for maximizing classifier efficacy while minimizing time or effort.* Such systems would be greatly valuable, leading to improved classifiers in production, shorter times to deployment, and lower development costs (considering the substantial labor cost of highly trained data mining programmers who are already spending their time to develop such classifiers). As a short name suggestive of this concept, we shall use *Programmer's Active Learning (PAL)*—with the intention to include systems that target technically-skilled data miners, even where general programming skills are not actually required.

Is this asking for the moon? Yes. But we have actually gone to the moon and even Mars. So, let us consider together at this workshop on *Active Learning in Real-world Applications* what might be possible. Here are three key issues to seed the discussion: the space of actions available, the user interface, and the action selection mechanism.

Potential action space: First of all, we need to consider with an open mind what might be included in the space of potential actions available to the typical, technically capable data miner when faced with a novel classification task. We

[2] The catch-all term 'real-world' has been taken to mean many diverse things in different papers. Here we use it to denote a task for which one truly desires to get the best classification accuracy possible within some reasonable budget of effort or time—as opposed to merely trying to beat some baseline methods on a benchmark using a fixed model and simple feature representation, as is often the case. High levels of motivation were apparent in the $1M Netflix competition. In these settings, concerns about saving some CPU cycles drop away, and we consider many more options that may be available to us, sometimes even including re-framing the problem definition.

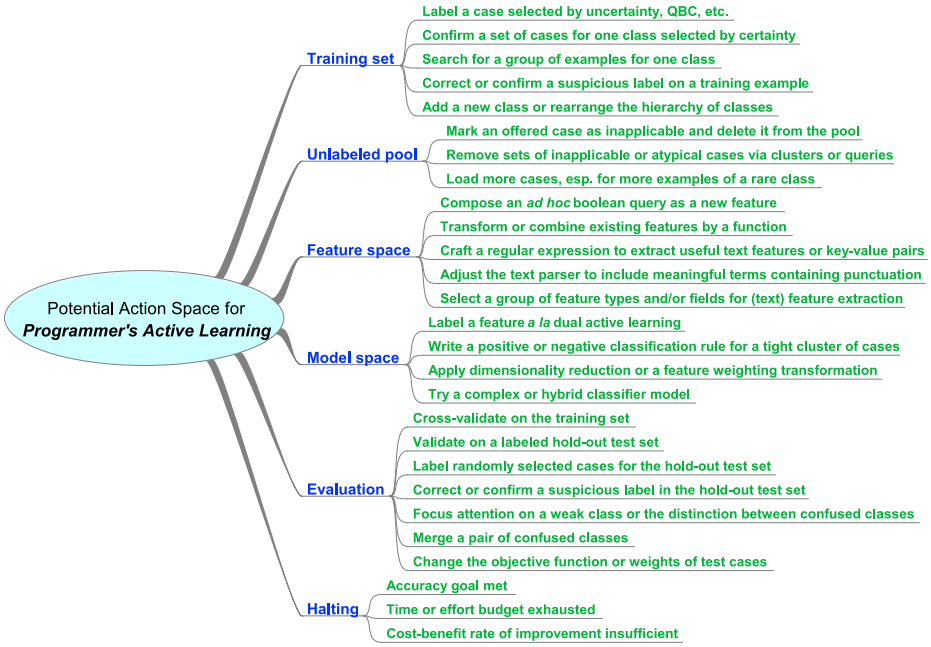

Fig. 1. Potential space of actions that a technically-capable data miner might choose from. Some of these could be monitored for cost-benefit improvements and even suggested under guidance from a future *Programmer's Active Learning* system.

have listed some ideas in Figure 1, considering a conventional classification task which might be binary, multi-class, or hierarchical. We illustrate a few examples:

1. Given a simple but powerful search box, a domain expert can often concoct a query that quickly identifies a group of cases that belong to one class. Even if the query results require some manual pruning, labeling this way can quickly grow the training set compared with labeling individual uncertain cases in a traditional active learning cycle.[7] Likewise, a PAL system could occasionally choose to employ *certainty* sampling for a particular small class in order to more quickly increase its training examples.
2. Notwithstanding the caveats of cross-validation under active learning sampling[2], by reviewing supposed errors from cross-validation, the user may be able to correct mistakes in the given labels. Corrections both purify the training data to simplify the task of the learning algorithm, as well as improve the apparent accuracy measured in the evaluation. A PAL system may focus on a class with apparently poor precision and ask the user whether some of the recalled 'false positive' cases are actually just mislabeled. Likewise for 'false negatives.'
3. Once an adequate training set is established, it may help to focus the domain expert on a weakly recognized class—showing either the labeled training ex-

amples thus far or the matching and near-matching cases from the pool—or else to focus their attention on a pair of classes that are often confused in cross-validation—showing the recognized and confused cases separately. With this focus on a subproblem, the domain expert may be able to come up with insightful ideas for additional predictive features. In a numerical domain, the user might enter some spreadsheet-like formulas to transform or combine existing features. In an image or video domain, a programmer-user might write some image processing code to extract a variety of features. In a text domain, the user may enter an *ad hoc* full-text query or a regular expression to extract new features from particular fields of the cases, e.g. in a domain of command-line inputs, it may be useful to provide a regular expression `\s--?(\w+)(=\S+)?` for extracting key-value arguments. Sometimes regular expressions or simple rewrite-rules can be authored to better normalize an input field treated as a nominal string value, such as to correct misspellings or fold needless variations in encoding. In technical text, one can sometimes adjust the set of characters that are considered word characters in order to capture additional meaningful terms; for example, on a past project involving technical support documents, we permitted some punctuation to be treated as word characters in order to expand the predictive terms extracted, such as 'MPE/XL', 'FDDI/9000', and 'HP/UX-11.1'.[5] The user may select different feature extractors for different combinations of text fields, e.g. bag-of-words-and-phrases for *title, abstract*, and *body* individually as well as concatenated, but do something else for *author, affiliation, classification,* and *references*. Or perhaps, seeing the confused cases and having little hope that the system could ever distinguish them reliably, one might just merge the classes, changing the objective for future PAL iterations.

4. One could try various machine learning models. Although this *model selection* task might be automated where it is just a matter of selecting among a closed list of models (e.g. C4.5, SVM, or Naive Bayes), sometimes on a hunch we data miners will concoct a complicated meta-model of ensembles, bagging, boosting, clustered features, feature selection, feature weighting, and dimension transformation. This requires a level of creativity and insight that only a skilled data miner can provide. However, a PAL system could support us with automated lesion studies, showing us which parts of the model matter most.
5. It can sometimes be useful to see various clusters of the pool cases. Upon seeing a pure cluster, a user might label its cases as training examples for one class, or form an ongoing rule to assign future matching cases to that class, hybridizing the model with rules. Or some clusters might be marked as not applicable for production and removed from consideration in the pool.
6. Given a labeled hold-out sample, the PAL system could periodically perform out-of-pool testing to determine whether the current classifier is satisficing for its intended use, especially if given an accuracy threshold. It may also choose to spend some user time building up or vetting the labeled hold-out sample.

Although we may ponder a wide space of actions, early PAL systems will support just the more basic actions that they can easily handle. Some actions could be available within the user-interface, even though they fall beyond the scope of the PAL system to estimate their value, such as choosing to enable the use of hidden features that would involve costs for measurements [12] or reworking the subroutine that is used to feed the pool of unlabeled cases [6]. More generally, at any time one may interactively re-define the problem substantially: adding or deleting classes, or changing their target concepts.[7] Judging the suitability of recommending such objective-changing actions would probably be considered beyond the scope of any realistic PAL algorithm...at least in the near term.

User-interface generality: Bespoke systems for active learning display a selected case to the domain expert and solicit the correct class label via a multiple-choice widget of some kind. Most of the additional actions proposed above can be supported by simple user-interface widgets. For example, text input boxes can accept full-text queries, regular expressions, or snippets of Perl or Python code to be eval()'d as features or rules; the GUI may need to provide feedback on the correctness and efficacy of the proposed snippets, which is easily done. Spreadsheets and SQL clients offer more sophisticated GUIs for composing and checking formulas and queries. In the most general case requiring arbitrary programming, the active learning control component may need to operate in a separate process from the user's edit-compile-execute loop, which repeatedly launches new processes with augmented code. One could imagine a plug-in for this within the Eclipse programming IDE that leverages Java's ability to dynamically load new code within a running process, so that large datasets and context need not be re-loaded. Thus, even incremental programming enhancements to the learning system could be accommodated within a running PAL system.

Cost-benefit estimation: Next, we need to consider how an algorithm might choose among the many potential actions. Just as *multi-arm bandit algorithms* need to measure and predict the reward for each separate action, so too for a PAL system. Despite the difficulties with cross-validation in real-world settings [2], some form of automated evaluation is needed to estimate the classification accuracy benefit of different potential actions, possibly via automated randomization studies or lesion studies on recent labels or features contributed by the programmer-user [6]. Note that to make a choice we do not need to accurately estimate the absolute expected benefit of each choice, but only to be often correct about their relative expected benefits (or the argmax). In some cases, there may be no way to predict the benefit that the user can provide by some actions, e.g. scrutinizing a pair of confused classes to propose new distinguishing features; but in some of these cases, at least an upper bound on the benefit can be estimated to better support the user in making informed choices.

Unlike traditional multi-arm situations, the cost for each action varies substantially and is unknown; it is usually much faster to label some cases than to develop a new regular expression feature. Thus, the PAL system will need to

track the time spent (or some other measure of effort) for each action taken by the programmer-user. Given approximate measurements of the costs and benefits over many trials, a PAL system could help guide the data miner, say, away from identifying additional stopwords or removing distracting text blocks, and toward other actions that historically have shown the best return on investment. Some of these learned parameters could carry across to future tasks: *life-long meta-learning*.

3 Conclusion

Just as traditional active learning can make domain experts many times more productive at labeling, the availability of a growing, labeled data set and a classification accuracy objective could be used in a semi-automated way to bring great improvements in programmer productivity for data mining experts. Imagine a future Eclipse plug-in environment that helps the programmer focus on the most useful tasks and features to add, analogous to the Mylyn task-focusing plug-in. There are many details to figure out and alternative avenues to explore, which provides excellent opportunities for many Ph.D. theses. What is here envisioned is quite broadening and ideally will lead to a variety of different research projects bringing worthwhile contributions.

One practical concern for publishing research in this area is whether reviewers, who are used to getting simulated active learning studies on several baseline methods and datasets, will grow to accept submissions in this area, since it is much more difficult to judge the true benefit of such a complex, interactive system. User-experience studies suffer comparatively weak persuasiveness and scientific reproducibility. Such work will need to clearly articulate the potential high value of success in this area and demonstrate substantial benefits. Incremental gains can eventually lead to 'real-world' success—achievements that matter to people faced with these tasks.

References

1. S. Arora and E. Nyberg. Assessing benefit from feature feedback in active learning for text classification. In *Proceedings of the 15th Conference on Computational Natural Language Learning*, CoNLL '11, pages 106–114, 2011.
2. J. Attenberg and F. Provost. Inactive learning? Difficulties employing active learning in practice. *SIGKDD Explor. Newsl.*, 12:36–41, March 2011.
3. L. Candillier. Nomao challenge. In *ALRA: Active Learning in Real-world Applications*, ECML-PKDD '12 Workshop, 2012. http://us.nomao.com/labs/challenge.
4. G. Druck, B. Settles, and A. McCallum. Active learning by labeling features. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1*, EMNLP '09, pages 81–90, 2009.
5. G. Forman. *Computational Methods of Feature Selection*, chapter Feature Selection for Text Classification, pages 257–276. Chapman and Hall/CRC Press, 2007.
6. G. Forman, E. Kirshenbaum, and S. Rajaram. A novel traffic analysis for identifying search fields in the long tail of web sites. In *Proc. 19th Int'l World Wide Web Conf.*, WWW'10, pages 361–370, 2010.

7. G. Forman, E. Kirshenbaum, and J. Suermondt. Pragmatic text mining: minimizing human effort to quantify many issues in call logs. In *Proc. 12th ACM SIGKDD conf. on Knowledge Discovery and Data mining*, KDD'06, pages 852–861, 2006.
8. I. Guyon, G. Cawley, G. Dror, and V. Lemaire. Results of the active learning challenge. *Journal of Machine Learning Research*, 16:19–45, 2011. AISTATS 2010 Workshop on Active Learning and Experimental Design.
9. R. Hu. *Active learning for text classification.* PhD thesis, Dublin Institute of Technology, 2011.
10. A. Krishnakumar. Active learning literature survey, 2007. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.116.4937.
11. H. Raghavan, O. Madani, and R. Jones. Active learning with feedback on features and instances. *J. Mach. Learn. Res.*, 7:1655–1686, Dec. 2006.
12. M. Saar-Tsechansky, P. Melville, and F. Provost. Active feature-value acquisition. *Manage. Sci.*, 55:664–684, April 2009.
13. B. Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009.
14. B. Settles. From theories to queries: Active learning in practice. *Journal of Machine Learning Research*, 16:1–18, 2011. AISTATS 2010 Workshop on Active Learning and Experimental Design.
15. B. Settles. *Active Learning*, volume 6 of *Synthesis Lectures on Artificial Intelligence and Machine Learning.* Morgan and Claypool, June 2012.
16. V. S. Sheng, F. Provost, and P. G. Ipeirotis. Get another label? Improving data quality and data mining using multiple, noisy labelers. In *Proc. 14th ACM Int'l Conf. on Knowledge Discovery and Data mining*, KDD '08, pages 614–622, 2008.
17. V. Sindhwani, P. Melville, and R. D. Lawrence. Uncertainty sampling and transductive experimental design for active dual supervision. In *Proc. 26th Int'l Conf. on Machine Learning*, ICML'09, pages 953–960, 2009.
18. L. Zhao, G. Sukthankar, and R. Sukthankar. Incremental relabeling for active learning with noisy crowdsourced annotations. In *Proceedings of the 3rd IEEE International Conference on Social Computing*, 2011.
19. Y. Zheng, S. Scott, and K. Deng. Active learning from multiple noisy labelers with varied costs. In *Proceedings of the 2010 IEEE International Conference on Data Mining*, ICDM '10, pages 639–648, 2010.