A Hybrid Symbolic-Subsymbolic Transformation Architecture as a Whiteboxing Tool for Machine Learning Systems Targeted at Non-Expert Users

Christian Leichsenring
CITEC
Bielefeld University
Bielefeld, Germany
cleichsenring@techfak.uni-bielefeld.de

Marc Hanheide
School of Computer Science
University of Birmingham
Birmingham, UK
marc@hanheide.de

Thomas Hermann
CITEC
Bielefeld University
Bielefeld, Germany
thermann@techfak.uni-bielefeld.de

ABSTRACT

Hybrid transformation architectures are systems that translate between the world of the subsymbolic in which most commonly used machine-learning techniques live and the world of symbolic rules that have largely fallen out of favor for intelligent systems but are still much closer to human language than subsymbolic systems are. Rule extraction allows the conversion of subsymbolic models to sets of rules while the opposite direction is called knowledge insertion.

In this paper we argue why we think that this can not only allow experts to better understand why their systems behave the way they do as has been the case for traditional applications of these techniques, but that they can also facilitate the leap of learning systems to non-expert end-users who might in future profit from such systems in their daily lives to manage the amount of personalized data that pervasive computing brings about.

We subsequently present a prototype system that will help us answer these questions in future research and discuss lines along which the evaluation of the presented ideas might proceed.

Categories and Subject Descriptors

I.5.0 [**Pattern Recognition**]: General; I.2.6 [**Learning**]: Knowledge acquisition

General Terms

Human Factors

Keywords

Rule extraction, knowledge discovery, hybrid transformation architectures, HCI, context-awareness

1. INTRODUCTION

With the advent of pervasive computing, ambient intelligence and the ever increasing amount of information streams that need personalized filtering, systems using machine learning are going to appear increasingly in everyday applications, affecting users that never were knowingly in contact with such adaptive systems. While this is no problem for systems that "just work" such as web searches, fingerprint scanners or face recognition in photo applications, it might pose a problem for systems that try to tackle problems that have a less comfortable ratio of problem complexity and available training data. Even the long-standing application and prime example of machine learning for end-users, the Bayes mail filtering, suffers somewhat from its unpredictability and attempts have been made to increase the comprehensibility of the filter decisions [3]. Context-aware applications on mobile platforms or in smart rooms, tailored to each user's individual behavior patterns, including custom locations, daily schedules, preferences, personality traits and biometric features such as gait, present a much more difficult class of challenges as finding a one-size-fits-all kind of solution might not in general be possible. Especially since the amount of feedback from the user for supervised learning is limited if the assistance of the system is not supposed to be more of an impediment than it can actually contribute to e. g. reduce interruptions of the user or save time through context-aware services. A system, however sophisticated, is useless when users switch off the adaptivity feature because it does not seem to help and they simply do not understand why it does the things it does or how they can make it do the things they want it to do.

The best-performing machine learning algorithms are subsymbolic in nature, adjusting weights, probabilities or high-dimensional vectors in order to duplicate complex real-world patterns. It is illusory to expect end-users to interpret or even purposefully change these parameters. Even experts in these models are hardly able to do so. Symbolic, rule-based systems on the other hand, albeit still being artificial and often hard to grasp, are much more similar to natural language and actually find application in tasks such as simple mail filtering or smart playlists.

We therefore suggest to address the foreboding acceptance problems of black-box systems governing important aspects of people's personal lives by whiteboxing them using symbolic, comprehensible representations and by allowing users to modify the extracted rules to modify the behavior of the learned model, giving them a sense and means of control way beyond (but still including) the passive learning-by-example capabilities such systems usually have. The idea of extracting rules from subsymbolic models is not new (cf. Section 1.2). What is new, however, is a full bi-directional symbolic-subsymbolic workflow for context-aware systems

Technical Report 2011 Bielefeld, Germany

that are targeted at non-expert users with only a limited amount of computer skills.

The proposed workflow is not only supposed to increase user acceptance but also to actually increase the model's predictive power because users naturally have a unique insight into their individual preferences and habits. They therefore are excellent domain experts in a sense, whose potential has to be unlocked in an especially unobtrusive and easy-to-use way though.

As an experimentation testbed and proof-of-concept system we wished for a context-aware system that has a differentiated interruptibility status, is unobtrusive in that it integrates nicely with every-day computer usage and translates well to other applications such as mobile platforms. Therefore we chose the availability status of an instant messenger as classes that are to be learned by the subsymbolic classifier. As sensors and data sources, only built-in capabilities of standard laptop computers were used.

1.1 Workflow

In order to use the prototype system called *Umber*, users first have to generate training data by setting their instant messenger (IM) status correctly and otherwise using their computer normally. Umber will record this status along with the sensor and system data listed in Section 2.1. After a period of passively recording data, Umber can be switched into a mode where it will use the collected data to try to predict the IM status of the users. The users can correct false predictions by resetting the IM status.

To understand the reasons for the predictions of the system, users can generate rules from the model Umber uses. For a given model an arbitrary number of rule sets can be generated and be inspected independently.

The ability of the user to change the underlying model through *knowledge insertion* by changing the generated rule set is just being implemented and therefore not functional yet.

1.2 Related Work

A good survey on rule extraction algorithms is given by Huysmans et al. [9]. They also list several properties of such algorithms, two very important ones for our application being *accuracy* and *fidelity*. Accuracy is the predictive power of the extracted rules regarding new data while fidelity signifies the similarity between the prediction behavior of the rule set and the black box model it had been extracted from. While for many applications accuracy is a prime objective, we primarily wish to balance comprehensibility and fidelity.

Wermter and Sun [15] coined the term "hybrid transformation architecture" for systems that translate from the subsymbolic to the symbolic domain or back. They also presented a comprehensive taxonomy of such systems.

In the field of data mining, rule extraction techniques are often equated with the term "knowledge discovery in data(-bases)" (KDD) [12, 7, 13]. In this field though, the systems are mainly targeted at highly skilled experts who are willing to put a lot of effort into gaining insight into hidden structures in data. This is quite the opposite of the casual non-expert users who mainly want the technology they use not to bother them.

Predicting user status and interruptibility for desktop usage has been shown to be possible before [6].

In related fields this has also been demonstrated practically, for example in ambient intelligence [14, 2] and on smartphone platforms [5, 16].

2. IMPLEMENTATION

2.1 Features

As data sources we used the microphone, camera, system time, the time since the last mouse or keyboard activity and information about the currently active window and application. Umber relies on the X Window System for this latter set of features.

The active window features are the only string features. To fit into the model we used, they were broken up into tokens at spaces and punctuation characters and then converted to numeric features by determining the frequency of each token in the training data and picking the 3-quantile of the most frequent tokens. Then each selected token became a new binary feature, signifying the token's presence.

All features were recorded every five seconds. For the audio and video features we additionally computed averages over the last $n = 5$ time steps.

The five labels of the data used for classification that were directly derived from the instant messenger status were: "free for chat", "online", "away", "unavailable" and "busy".

2.2 Model Generation

For classification we used a support vector machine with an RBF kernel from LibSVM [1], wrapped by the Weka toolkit [8]. We did a test using a small preliminary data set of sample size 1531. We left out the time features since they cannot contain useful information with so few and short sessions and might lead to a undeservedly good result. We did a random $^2/_3$-split into a training and a test data set and performed a hyperparameter grid optimization on the training data with a 5-fold cross validation using LibSVM's grid tool. On the test set, the generalization accuracy was 98.2% with parameters $C = 512$ and $\gamma = \frac{1}{8192}$

2.3 Rule Extraction

For choosing a rule extraction algorithm we first defined a number of requirements. In particular, we 1) need easily understood rules, 2) prioritize fidelity (model prediction) over accuracy (data prediction), 3) wish independence from the learning algorithm used (such a method is also called *pedagogical*), 4) would like to be able to try regression as well as classification, 5) need a certain stability of the generated rules (i. e. small changes in the model lead to small changes in the rule set), but 6) on the other hand we would like to let the users generate a forest of different decision trees from which the one with the best comprehensibility or fidelity can be selected. And finally we 7) would like to be able to generate different types of rules (e. g. if-then, M-of-N, fuzzy rules) because some types might be easier to understand than others.

Discussing the advantages and disadvantages of all available rule extraction techniques goes beyond the scope of this paper but again, Huysmans et al. [9] provide a good overview of many of the requirements stated above. The two methods that best fit our requirements were TREPAN [4] – which works somewhat similar to other decision tree algorithms such as C4.5 but prominently uses a very sophisticated way to estimate missing data points from the underlying trained model – and G-REX [10] which uses evolutionary programming to find its rule sets. Neither one perfectly fits all requirements though. TREPAN is largely deterministic and thus does not generate alternatives by itself. G-REX on the other hand does so quite naturally and can even be encouraged to do so [11]. Because of the stochastic algorithm, it conversely has

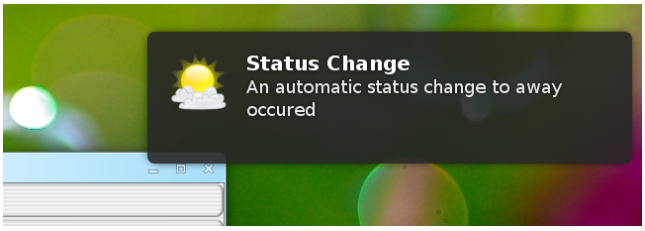

Figure 1: Desktop notifications are used to unobtrusively inform the user about the status change. The icon symbolizes the new status.

an increased instability of rules though, beyond the normal informational instability. Regarding most other criteria, both techniques perform equally well. G-REX, however, also has the advantage of being able to produce different types of rules. We therefore chose G-REX over TREPAN.

The decision trees G-REX generates can be tuned to be more accurate or less complex as desired by changing the parameters of the benefit function that selects the best trees in each generation.

2.4 Interface

The main user interaction for data collection is done using a conventional instant messenger. As instant messaging platform, the open XMPP-based Jabber protocol was used. Thanks to the XMPP remote control extension[1] we can integrate with any client supporting this extension without having to change any client code itself.

The user is informed about automatic status changes from the system using desktop notifications[2] (Figure 1). Negative feedback can be given by just resetting the status in the client.[3]

The GUI for controlling the data collection activity and the automatic setting of the status by the system can be seen in Figure 3. This is also where the rule extraction is initiated and the rule viewer and editor is called. The latter is shown in Figure 2.

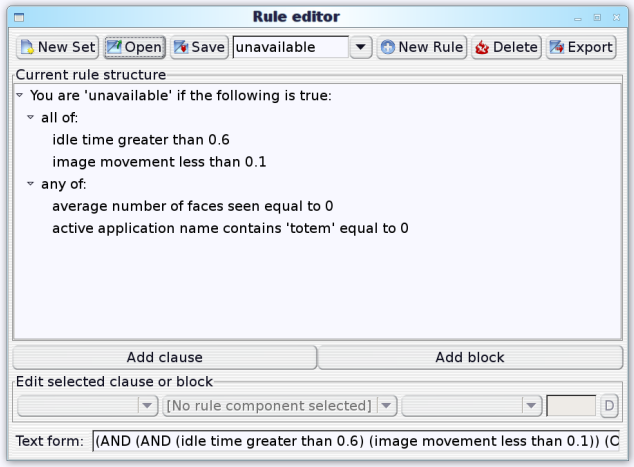

Figure 2: The current draft of the rule editor. It serves to present the generated rules to the users and allows them to edit them. The knowledge insertion back into the subsymbolic model is currently a work in progress, as are the usability of the GUI and the presentation of the features.

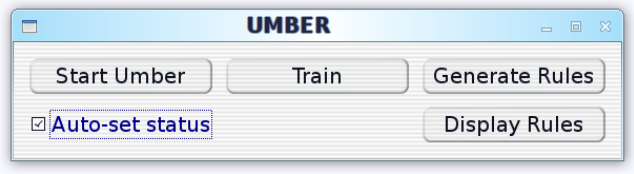

Figure 3: The main window of the GUI is deliberately kept simpel. It basically allows the user to start the recording, select whether or not to automatically set the Jabber status from the trained model and to generate and inspect rule sets.

3. DISCUSSION

While building the presented system is an important step, it is only the first step in answering the more fundamental research questions that we would like to briefly present in the following.

The hypotheses we would like to test with Umber regarding increased usability through hybrid transformation architectures are the following: i) Non-expert users can understand the rules produced from a stochastic adaptive system (and can do so better than the original system's parameters itself). ii) Such inspection possibilities make them more comfortable with such systems. iii) They can manipulate such rule sets in a sensible way, thereby contributing their domain knowledge. iv) Such manipulation possibilities give them a sense of control over the system. v) This further enhances the acceptance of adaptive systems.

Being able to inspect and change learned models can also encourage users to provide training data that they might otherwise have discarded because they were afraid of spoiling the classifier. An example might illustrate this: a user gets a couple of spam emails, each of which contains the same set of rare words that the user knows not to be related to spam. Also, the user might know that these words did not occur in the prior mail correspondence but might very well occur one day. By classifying these spam mails as spam the user has to be afraid that the spam classifier will cling onto these rare words as very good indicators of spam, although the user knows this to be a false yet reasonable conclusion. If there was a possibility to see whether these false conclusions were in fact drawn and to fix this if it occurred, users would be freed from such distracting thought processes and would instead provide more potentially valuable training data to the classifier.

There are also some more technical hypotheses we need to test: i) It is practically possible to predict complex context relations such as a user's interruptibility with available sensors (with or without the intervention of the user through rule interfaces). ii) A practical amount of fidelity for comprehensible rule sets is possible. iii) Inserted knowledge from rule changes is reflected in re-extracted rules. iv) A practical amount of rule stability is possible.

[1] `http://xmpp.org/extensions/xep-0146.html`
[2] `http://www.galago-project.org/specs/notification/`
[3] On platforms supporting it, buttons on the desktop notification bubble itself can be used for user feedback.

To evaluate both these ranges of questions, we are planning a longitudinal study in which users accumulate enough data to train a practical classifier while they constantly monitor its performance as well as the applicability of generated rules. We want to ask users to fill in a minimal questionnaire daily and a more profound one at the end of the study. We will also record the way participants use the system.

A further aspect that needs to be addressed is the preferred presentation and manipulation interface. This includes the kind of rules that are extracted from the model (fuzzy rules for example are often said to be more comprehensible) as well as the user interface itself.

No presentation can compensate for a fundamental issue with this kind of generated rules though, which is the fact that the symbols used are low-level features meant to be interpreted by machines, not humans. A possible escape from this problem would be to use unsupervised learning to find structures in the data and then present recordings from typical points in these clusters to the user who can then label them. The hope is that this would mean that separate kinds of situations that are present in the data can be associated with their natural-language description and this can then be used as a more readily understood symbol that might represent not a single feature but an arbitrary combination of features, defined by the cluster centers.

Lastly, it is clear that generated rules can always be only a rough approximation of the underlying model or the incomprehensibility of the rules would not be far from the obscurity of the stochastic model itself. This also explains why rule-based classifiers are no real alternative to our approach: in order to decently model the necessary patterns, their complexity would exceed the comprehensible amount of rules by far. The change of symbols we just described to represent the same relation in the data would also be very difficult with purely rule-based systems.

4. CONCLUSION

We presented the idea of increasing the acceptance of black-box adaptive systems by extracting symbolic rules from the models that allow non-expert users to understand the most important patterns governing such a model's behavior and modifying or inserting custom patterns reflecting the user's specific priorities and personal behavior.

We introduced a prototype system that implements this in the context of interruptibility detection using an instant messenger whose status can be used to train a model with sensors available on most laptop computers and that can subsequently predict a user's IM status using the recorded data. A rule inspection interface was added to this system using the G-REX framework.

Finally, we discussed the kind of research questions that can be addressed with this system and suggested ways to test them. We also briefly brought up some issues such non-expert user transformation architectures between the subsymbolic and symbolic domains might have and ways to ameliorate them.

With this work we hope to inspire others to also consider symbolic-subsymbolic transformation architectures as a tool for user friendly adaptive systems and ultimately to advance the use of machine learning in personalized everyday scenarios such as smart assistants who can explain themselves and easily understood smart artifacts.

5. REFERENCES

[1] C.-C. Chang and C.-J. Lin. *LibSVM: A Library for Support Vector Machines*, 2001. `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[2] B. Chikhaoui, S. Wang, and H. Pigot. A frequent pattern mining approach for adls recognition in smart environments. *Advanced Information Networking and Applications, International Conference on*, 0:248–255, 2011.

[3] W. W. Cohen. Learning rules that classify e-mail. In *AAAI Spring Symposium on Machine Learning in Information Access*, volume 18, page 25, 1996.

[4] M. W. Craven and J. W. Shavlik. Extracting tree-structured representations of trained networks. *Advances in neural information processing systems*, 8:24–30, 1996.

[5] A. Dekel and D. Nacht. Minimizing mobile phone disruption via smart profile management. In *Proceedings of the 11th International Conference on Human-Computer Interaction with Mobile Devices and Services*. ACM, Sept. 2009.

[6] J. Fogarty, S. E. Hudson, C. G. Atkeson, D. Avrahami, J. Forlizzi, S. Kiesler, J. C. Lee, and J. Yang. Predicting human interruptibility with sensors. *ACM Trans. Comput.-Hum. Interact.*, 12:119–146, Mar. 2005.

[7] M. M. Gaber. *Scientific data mining and knowledge discovery – principles and foundations*. Springer, Heidelberg [u.a.], 2010.

[8] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: An update. *SIGKDD Explorations*, 11(1):10–18, 2009.

[9] J. Huysmans, B. Baesens, and J. Vanthienen. Using rule extraction to improve the comprehensibility of predictive models. Technical report, Katholieke Universiteit Leuven, Faculty of Business and Economics, 2006.

[10] U. Johansson, R. König, and L. Niklasson. Rule extraction from trained neural networks using genetic programming. In *13th International Conference on Artificial Neural Networks*, pages 13–16, 2003.

[11] R. König, U. Johansson, and L. Niklasson. Finding the tree in the forest. In *2010 IADIS International Conference Applied Computing*, volume 135–142. IADIS Press, 2010.

[12] H. Liu and H. Motoda. *Feature selection for knowledge discovery and data mining*, volume 454. Springer, 1998.

[13] M. J. Pazzani. Knowledge discovery from data? *Intelligent systems and their applications, IEEE*, 15(2):10–12, 2000.

[14] D. Sanchez, M. Tentori, and J. Favela. Hidden markov models for activity recognition in ambient intelligence environments. *Mexican International Conference on Computer Science*, 0:33–40, 2007.

[15] S. Wermter and R. Sun, editors. *Hybrid Neural Systems*. Springer, 2000.

[16] S. Zulkernain, P. Madiraju, and S. I. Ahamed. A context aware interruption management system for mobile devices. In *Mobile Wireless Middleware, Operating Systems, and Applications*, pages 221–234. Springer Berlin Heidelberg, 2010.