27th International Joint Conference on Artificial Intelligence and the 23rd European Conference on Artificial Intelligence

July 13-19, 2018, Stockholm, Sweden

XAI 2018

Proceedings of the 2nd Workshop on

Explainable Artificial Intelligence

Edited by:

David W. Aha, Trevor Darrell,
Patrick Doherty and Daniele Magazzeni

Organization

David W. Aha
Naval Research Laboratory , USA

Trevor Darrell
UC Berkeley, USA

Patrick Doherty
Linköping University, Sweden

Daniele Magazzeni
King's College London, UK

Program Committee

David W. Aha
Naval Research Laboratory

Sambit Bhattacharya
Fayetteville State University

Or Biran
Columbia University

Tathagata Chakraborti
Arizona State Univeristy

Dustin Dannenhauer
National Research Council, Naval Research Laboratory

Michael Floyd
Knexus Research Corporation

Freddy Lecue
Accenture Labs

Ninghao Liu
Texas A&M University

Daniele Magazzeni
King's College London

Fabio Mercorio
University of Milano Bicocca

Matthew Molineaux
Knexus Research Corporation

Simon Parsons
King's College London

Mark Roberts
Naval Research Laboratory

Liz Sonenberg
Melbourne University, Department of Information Systems

Mor Vered
Bar Ilan University

Nirmalie Wiratunga
The Robert Gordon University

Yezhou Yang
Arizona State University

Foreword

Explainable AI (XAI) systems embody explanation processes that allow users to gain insight into the system's models and decisions, with the intent of improving the user's performance on a related task. For example, an XAI system could allow a delivery drone to explain (to its remote operator) why it is operating normally and the situations for when it will deviate (e.g., avoid placing fragile packages on unsafe locations), thus allowing an operator to better manage a set of such drones. Likewise, an XAI decision aid could explain its recommendation for an aggressive surgical intervention (e.g., in reaction to a patient's recent health patterns and medical breakthroughs) so that a doctor can provide better care. The XAI system's models could be learned and/or hand-coded, and be used for a wide variety of analysis or synthesis tasks. However, while users of many applications (e.g., related to autonomous control, medical, or financial decision making) require understanding before committing to decisions with inherent risk, most AI systems do not support robust explanation processes. Addressing this challenge has become more urgent with the increasing reliance on learned models in deployed applications. This raises several questions, such as: how should explainable models be designed? How should user interfaces communicate decision making? What types of user interactions should be supported? How should explanation quality be measured? These questions are of interest to researchers, practitioners, and end-users, independent of what AI techniques are used. Solutions can draw from several disciplines, including cognitive science, human factors, and psycholinguistics.

This is the second XAI workshop, the first one was held at IJCAI-17. The XAI workshops provide a forum for learning about exciting research on interactive XAI methods, highlighting and documenting promising approaches, and encouraging further work, thereby fostering connections among researchers interested in AI (e.g., causal modeling, computational analogy, constraint reasoning, intelligent user interfaces, ML, narrative intelligence, planning) human-computer interaction, cognitive modeling, and cognitive theories of explanation and transparency. The XAI workshops focus on agent explanation problems, motivated by human-machine teaming needs. This topic is of particular importance to (1) deep learning techniques (given their many recent real-world successes and black-box models) and (2) other types of ML and knowledge acquisition models, but also (3) application of symbolic logical methods to facilitate their use in applications where supporting explanations is critical.

This year the XAI workshop received an excellent response in terms of submissions and participation. The program included five invited talks, lightning talks for the 30 accepted papers, a summary presentation of events held at the recent ICAPS-18 Explainable AI Planning Workshop (XAIP), and one poster session.

We would like to thank the authors who submitted their papers and all workshop attendees. Moreover, we sincerely thank the program committee for their great work in reviewing the papers.

David W. Aha, Trevor Darrell, Patrick Doherty, Daniele Magazzeni.
July 2018

Contents

Hybrid Data-Expert Explainable Beer Style Classifier
Jose M. Alonso, Alejandro Ramos-Soto, Ciro Castiello, and Corrado Mencar **1**

Explanations for Temporal Recommendations
Homanga Bharadhwaj and Shruti Joshi **6**

Towards Providing Explanations for AI Planner Decisions
Rita Borgo, Michael Cashmore, and Daniele Magazzeni **11**

Human-Aware Planning Revisited: A Tale of Three Models
Tathagata Chakraborti, Sarath Sreedharan, and Subbarao Kambhampati **18**

Explanatory Predictions with Artificial Neural Networks and Argumentation
Oana Cocarascu, Kristijonas Čyras, and Francesca Toni **26**

ScenarioNet: An Interpretable Data-Driven Model for Scene Understanding
Zachary Daniels and Dimitris Metaxas **33**

Explaining Deep Adaptive Programs via Reward Decomposition
Martin Erwig, Alan Fern, Magesh Murali, and Anurag Koul **40**

Generating Natural Language Explanations for Visual Question Answering using Scene Graphs and Visual Attention
Shalini Ghosh, Giedrius Burachas, Arijit Ray, and Avi Ziskind **45**

Interpretable Self-Labeling Semi-Supervised Classifier
Isel Grau, Dipankar Sengupta, María Matilde García Lorenzo, and Ann Nowe **52**

How Explainable Plans can Make Planning Faster
Antoine Gréa, Laetitia Matignon, and Aknine Samir **58**

A Qualitative Analysis of Search Behavior: A Visual Approach
Ian Howell, Robert Woodwar, Berthe Choueiry, and Hongfeng Yu **65**

Toward Learning Finite State Representations of Recurrent Policy Networks
Anurag Koul, Sam Greydanus, and Alan Fern **72**

Interpretable Neuronal Circuit Policies for Reinforcement Learning Environments
Mathias Lechner, Ramin M. Hasani, and Radu Grosu **79**

Exploring Explainable Artificial Intelligence and Autonomy through Provenance
Crisrael Lucero, Braulio Coronado, Oliver Hui, and Douglas Lange **85**

Explore, Exploit, and Explain: Personalizing Explainable Recommendations with Bandits
James McInerney, Benjamin Lacker, Samantha Hansen, Karl Higley, Hugues Bouchard, Alois Gruson, and Rishabh Mehrotra **90**

Declarative Description: The Meeting Point of Artificial Intelligence, Deep Neural Networks, and Human Intelligence
Zoltán Á. Milacski, Kinga Bettina Faragó, Áron Fóthi, Viktor Varga, and Andras Lorincz **97**

Assisted and Incremental Medical Diagnosis using Explainable Artificial Intelligence
Isaac Monteath and Raymond Sheh **104**

From Context Mediation to Declarative Values and Explainability
Grzegorz J. Nalepa, Martijn van Otterlo, Szymon Bobek, and Martin Atzmueller **109**

Local Interpretable Model-Agnostic Explanations of Bayesian Predictive Models via Kullback-Leibler Projections
Tomi Peltola **114**

Explanatory Masks for Neural Network Interpretability
Lawrence Phillips, Garrett Goh, and Nathan Hodas **119**

Transparency Communication for Reinforcement Learning in Human Robot Interactions
David Pynadath, Ning Wang, and Michael Barnes **123**

Analyzing Hypersensitive AI: Instability in Corporate-Scale Machine Learning
Michaela Regneri, Malte Hoffmann, Jurij Kost, Niklas Pietsch, Timo Schulz, and Sabine Stamm **130**

A Survey of Interpretability and Explainability in Human-Agent Systems
Avi Rosenfeld and Ariella Richardson **137**

A Symbolic Approach to Explaining Bayesian Network Classifiers
Andy Shih, Arthur Choi, and Adnan Darwiche **144**

Towards a Taxonomy for Interpretable and Interactive Machine Learning
Elio Ventocilla, Tove Helldin, Maria Riveiro, Juhee Bae, Veselka Boeva, Göran Falkman, and Niklas Lavesson3 **151**

Explainable Security
Luca Viganó and Daniele Magazzeni **158**

Contrastive Explanations for Reinforcement Learning in terms of Expected Consequences
J.S. van der Waa, Jurriaan Van Diggelen, Karel van den Bosch, and Mark Neerincx **165**

Show Me What You've Learned: Applying Cooperative Machine Learning for the Semi-Automated Annotation of Social Signals
Johannes Wagner, Tobias Baur, Dominik Schiller, Yue Zhang, Björn Schuller, Michel Valstar, and Elisabeth André **171**

Learning Attributions Grounded in Existing Facts for Robust Visual Explanation
Yulong Wang, Xiaolin Hu, and Hang Su **178**

Explicating Feature Contribution using Random Forest Proximity Distances
Leanne S. Whitmore, Anthe George, and Corey M. Hudson **183**

Hybrid Data-Expert Explainable Beer Style Classifier

Jose M. Alonso[1], **Alejandro Ramos-Soto**[1,3], **Ciro Castiello**[2], **Corrado Mencar**[2]
[1] Centro Singular de Investigación en Tecnoloxías da Información (CiTIUS),
Universidade de Santiago de Compostela, Santiago de Compostela, Spain
[2] Department of Informatics, University of Bari "Aldo Moro", Bari, Italy
[3] Department of Computing Science, University of Aberdeen, Aberdeen, UK
{josemaria.alonso.moral,alejandro.ramos}@usc.es, {ciro.castiello,corrado.mencar}@uniba.it

Abstract

This paper presents a hybrid data-expert approach for building eXplainable Artificial Intelligence (XAI) systems. It combines an opaque machine learning system with several interpretable systems to build a whole XAI system, i.e., a system which provides users with a good interpretability-accuracy trade-off but also with explanation capabilities. First, the opaque system acts as an "oracle" which finds out the most plausible output. Then, the simplest interpretable system carrying out the same output is selected. Finally, a textual explanation of the given output is generated, which emerges as an automatic interpretation of the inference process carried out by the selected interpretable system. The textual explanation is built by applying a Natural Language Generation Approach. The proposal is validated in a real use case related to beer style classification.

1 Introduction

The recent success of many AI applications into real-world usage has triggered some critical voices beyond technological issues. Since most AI applications interact with humans, ethical and legal issues become essential. For example, *Explanation* is highlighted in the ACM Code of Ethics [US Public Policy Council, 2017] as a basic principle in the search for "Algorithmic Transparency and Accountability". Moreover, the new European General Data Protection Regulation (GDPR), which took effect in May 2018, is already fostering a discussion about the meaning of the "right to explanation" [Goodman and Flaxman, 2016; Wachter *et al.*, 2017].

Nowadays, companies demand a new generation of eXplainable AI (XAI) systems, i.e., AI systems ready to explain their automatic decisions in a human-like fashion. Moreover, explainability has always been a relevant issue in the research field related to expert and knowledge-based (KB) systems [Buchanan and Shortliffe, 1984; Gaines and Boose, 1988]. It has also been addressed thoroughly for developing decision-support and recommendation systems [Tintarev and Masthoff, 2015]. The Machine Learning (ML) community has usually been more focused on accuracy than explainability, yet researchers are more and more aware of the need of considering explainability when designing ML algorithms [Biran and Cotton, 2017].

We believe that looking for further synergies between the KB and ML research communities is the right way to advance towards the desired new generation of XAI systems. This, in turn, will make AI more accessible to people. In this paper, we introduce a new hybrid (KB+ML)-based approach for XAI. First, an opaque ML system (i.e., an ML system without explanatory capabilities but endowed with high accuracy) points out the most plausible output. Then, we look for the simplest explanation associated to such output, among all the potential explanations provided by a pool of interpretable systems with different interpretability-accuracy trade-offs. Next, an intelligent agent first makes an automatic interpretation of the inference carried out by the selected system (i.e., the system with the best interpretability-accuracy trade-off) and then translates it into human-like text by means of natural language generation (NLG) technology.

The rest of the paper is organized as follows. Section 2 presents the proposed hybrid XAI approach. Section 3 goes in depth with a use case where the goodness of the proposal is evaluated by humans in an on-line survey. Finally, Section 4 remarks the main points of the study and pinpoints future work.

2 The Hybrid XAI Approach

We developed an XAI system combining ML-based and KB-based systems, with the aim of producing very accurate classifications together with their related explanations (see Fig. 1). First of all, we gather all the available information (data and expert knowledge) related to the classification problem under consideration. Then, we build the components of our XAI classifier in an off-line stage as follows. On the one hand, we build a very accurate yet non-interpretable ML-based classifier. This classifier acts as an "oracle" to guide the classification procedure.

In the use case described in Section 3 we tested two supervised learning methods:

- Multilayer Perceptron (MP) [Rosenblatt, 1961] is a feed-forward neural network for classification. It is known for producing very accurate but opaque classifiers.
- Random Forest (RF) [Breiman, 2001] is an ensemble

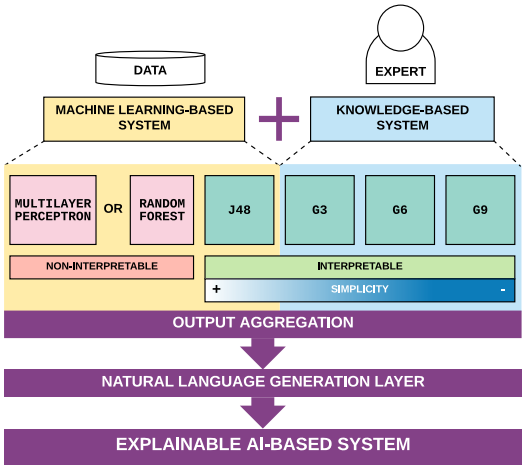


Figure 1: Hybrid Data-Expert XAI system.

learning method that creates a combination of decision trees (C4.5) [Quinlan, 1993]. Even though the single classifiers are deemed as interpretable, their random combination is hardly interpretable.

On the other hand, we build four interpretable classifiers:

- One ML-based classifier: J48 is the Weka [Witten *et al.*, 2016] method implementing C4.5.
- Three KB-based classifiers: We focus on methods supported by fuzzy sets theory (FST) [Zadeh, 1965] which provides a suitable grounding for interpretability issues [Alonso *et al.*, 2015]. Namely, fuzzy rule-based classifiers that are generated with GUAJE [Pancho *et al.*, 2013], a tool implementing the Highly Interpretable Linguistic Knowledge (HILK) fuzzy modeling methodology [Alonso and Magdalena, 2011]. HILK is aimed at producing a good interpretability-accuracy trade-off thanks to the combination of expert and induced knowledge. We initialize fuzzy classifiers with 3, 6 or 9 linguistic terms for each fuzzy partition (the related systems are called G3, G6, and G9, respectively). Then, we apply the fuzzy decision tree method provided by GUAJE to generate interpretable fuzzy rules.

Then, given a data instance to be classified (on-line stage), the system decision is computed as follows:

1. Local selection. If the class selected by the "oracle" (ML class) is supported by at least one of the four interpretable classifiers, then it is taken as the output class. The related explanation is based on the simplest interpretable classifier which points at the same class.
2. Global selection. In case the local selection fails, then we compute the output class as the median of the outputs provided by all the 5 classifiers (notice that output classes are represented by ordered numbers). If the aggregated (median) class is supported by at least one of the 4 interpretable classifiers, then the procedure goes on as explained previously (1). Otherwise, explanation is based on the simplest interpretable classifier with the most frequent class.

Then, an NLG module produces the textual explanation associated to the output class. Notice that NLG is a well-known area inside the computational linguistics field [Gatt and Krahmer, 2018; Dale, 2017]. In Data-to-Text applications, NLG plays a key role for generating text that is readable and understandable by humans [Ramos-Soto *et al.*, 2016]. Many NLG systems have been developed on the basis of the generic methodology and the architectural framework proposed by Reiter and Dale [Reiter and Dale, 2000].

In practice, we apply the same NLG methodology to generate explanations related to fuzzy rule-based classifiers and to crisp decision trees. We use the open source software rLDCP [Conde-Clemente *et al.*, 2017], which is an R package based on FST and on the Computational Theory of Perceptions [Zadeh, 2001; 1999]. rLDCP assists engineers to generate Linguistic Descriptions of Complex Phenomena (LDCP) [Trivino and Sugeno, 2013]. The issue here consists in providing automatic interpretation for the specific phenomenon under consideration, that is the classification procedure of an interpretable classifier previously learnt from data. To this end, the phenomenon is first translated into data, then data are interpreted in terms of the granular linguistic model of phenomena (GLMP), and finally the results of interpretation are verbalized in natural language through the report generation procedure [Trivino and Sugeno, 2013].

The GLMP, standing at the core of the LDCP architecture, is a hierarchical network of perception mappings (PMs) and computational perceptions (CPs). Each CP is a tuple (A, W_A, R_A) where A is a vector of linguistic expressions, W_A and R_A are, respectively, vectors of validity and relevance degrees associated to A. Each PM is a tuple (U, y, g, T) where U is a vector of inputs; being numerical values at the lowest level of the hierarchy (where PMs are tagged as 1PM_i) and CPs in the rest of levels (where PMs are tagged as 2PM_j). y is the output CP, g is the aggregation function which translates inputs into output, and T is the text generation algorithm (in the simplest case just a text template).

Given a fuzzy rule-based classifier, the GLMP is built bottom-up as follows. In the lowest level of the hierarchy, there is a 1PM associated to each input variable. In the second level, there is a 2PM associated to each rule. In the top of the hierarchy, we find a 2PM which aggregates all the rules and gives the output class.

In the case of J48, we first translate the decision tree into a set of rules. Each rule represents one path in the tree from the root to a leaf. Then, the GLMP is built bottom-up following the same procedure already described for fuzzy rule-based classifiers.

3 Use Case

We focus on automatic classification of eight beer styles (Blanche, Lager, Pilsner, IPA, Stout, Barleywine, Porter, and Belgian Strong Ale) in terms of three attributes (color, bitterness and strength). The dataset is made up of 400 instances [Castellano *et al.*, 2017].

We applied 10-fold cross-validation as evaluation methodology (accuracy is measured in terms of classification rate). Table 1 reports the main values related to all the single clas-

sifiers considered in our approach. On the one hand, the performance of non-interpretable classifiers (MP and RF) is described in terms of average accuracy. On the other hand, to describe the performance of interpretable classifiers (J48, G3, G6 and G9) the accuracy values are accompanied by a basic interpretability metric (NR) related to the structural complexity of the model. NR counts the number of rules in a rule base (G3, G6, G9) and the number of branches in a tree (J48).

In addition, Table 2 summarizes the main results reported by the hybrid XAI approach presented in the previous section. For the two developed systems (MP-based XAI system and RF-based XAI system), the table includes first accuracy and gain with respect to the related non-interpretable system (MP or RF, respectively), considered alone. Then, it shows the percentage of data instances for which the explanation is based on each of the interpretable systems (J48, G3, G6, G9). For example, EXP-J48 represents the percentage of instances for which the explanation of the XAI system is based on the tree generated with J48.

As it can be appreciated in Tables 1 and 2, MP stands out as the most accurate single classifier. In addition, the XAI system increases accuracy regarding both MP and RF (gain is +0.5 and +1.25, respectively); with the MP-based XAI system achieving the highest accuracy (97% of classification rate). Notice that the XAI system developed in this paper overwhelms the systems developed in [Castellano *et al.*, 2017] from the point of view of accuracy: an expert system (8 rules with 3 inputs) with 81.25% of classification rate; and a fuzzy rule-based classifier (9 rules with 2 inputs) with 87.5% of classification rate. Moreover, our XAI system also offers to users the added-value of textual explanations.

The explanation procedure is as follows. Given a data instance (e.g., Color=2; Bitterness=18; Strength=0.049; see Fig. 2), once the interpretable classifier responsible for the explanation is selected (J48 in this example), we build the related textual explanation by running the rLDCP software with the corresponding GLMP (see Fig. 3, where the involved PMs and CPs are organized as described in the previous section). The lowest level in the hierarchy (1PMs) provides the user with textual information about the inputs (Color, Bitterness and Strength). The second level gives information about the activated rules (branches of the tree in case of J48). There is a textual description associated to the output given by each rule. It describes the way the inputs relate each other in the classification process. For example, the explanation given in Fig. 2 comes from the following rule (translated from the related branch in the tree generated by J48): "IF the beer color is smaller or equal than 6, its bitterness is smaller or equal than 26, and its strength is smaller or equal than 0.07 THEN Beer style is Blanche". The highest level in the hierarchy gives the final output as an aggregation of all the rules. Notice that in the case of J48 only one rule is activated for each data instance. However, several rules are usually activated at the same time in the case of fuzzy rule-based classifiers.

In order to assess the quality of the generated explanations, we set up an on-line survey with Google forms and announced it via email and via social networks. We showed five different samples in five consecutive screens to each subject. Notice that we first presented the decision made by the XAI system along with a related picture. Then, we included the related explanation along with additional details (see Fig. 2). Afterwards, we asked subjects to assess each explanation by rating the following statements on a five-level Likert scale: S1 (*The information above helps you understand the system's decision*); S2 (*The explanation is clear and does not need to be re-written in a clearer way*); S3 (*Providing decision along with explanation helps you to trust the system*).

Subjects with a Bachelor of Science (BSc) represented the main target of our interview. We also looked for subjects who were either native or non-native in English (we considered as native people living in English-speaking countries). That is why we asked assessors about their academic background, their level of English and their country of residence, at the end of the survey.

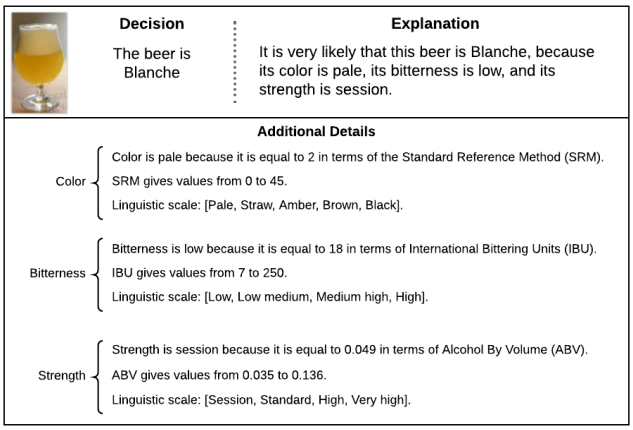


Figure 2: Example of output given by the XAI system.

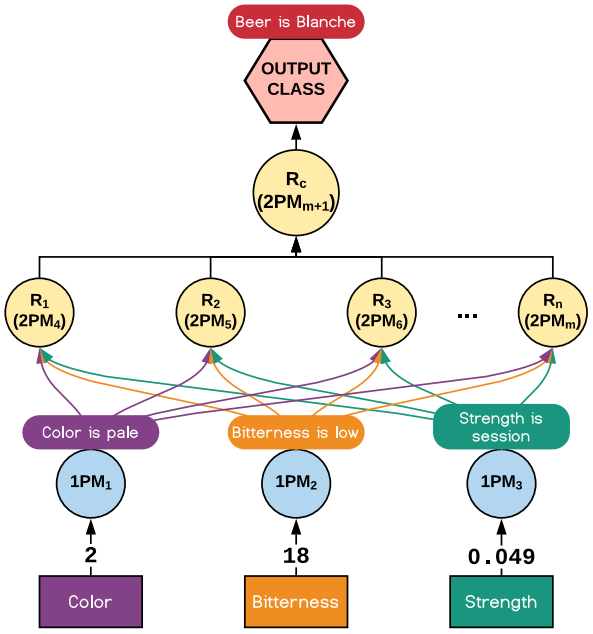


Figure 3: Automatic interpretation of the hybrid XAI output.

Table 1: Single AI systems (averaged results over 10-fold cross-validation).

	MP	RF	J48		G3		G6		G9	
	Accuracy	Accuracy	Accuracy	NR	Accuracy	NR	Accuracy	NR	Accuracy	NR
BEER	96.5	95	94.75	10.7	67.75	7.5	93.5	11.4	94.25	16.5

Table 2: Hybrid XAI system (averaged results over 10-fold cross-validation).

	MP-based XAI system						RF-based XAI system					
Datasets	**Accuracy**	**Gain**	**EXP-J48**	**EXP-G3**	**EXP-G6**	**EXP-G9**	**Accuracy**	**Gain**	**EXP-J48**	**EXP-G3**	**EXP-G6**	**EXP-G9**
BEER	97	+0.5	96.75	2	0.75	0.5	96.25	+1.25	98.5	0.75	0.5	0.25

Table 3: Summary of answers in the on-line survey (average score over all the 5 samples; min=1 and max=5).

	Group	S1	S2	S3	Subjects (%)
Category	**ALL**	3.8	3.35	3.96	100
Language	**English**	3.61	2.98	4.05	43.33
	Italian	3.95	3.64	3.9	56.67
Background	**BSc**	3.85	3.31	4.03	76.67
	No BSc	3.64	3.49	3.76	23.33

Table 3 summarizes the collected answers (the percentage of subjects per group is given in the last column). We got feedback from 60 anonymous subjects (76.67% with a BSc). 43.33% of subjects are native in English (or live in an English speaking country, such as UK or USA) while the rest has Italian as mother tongue. The first row in the table shows the average score for all the subjects (further details are given in Fig. 4). We consider that getting average score around 4 for both S1 and S3 means our XAI system is useful for the target audience. In addition, the lower score associated to S2 suggests that the system may be further enhanced by refining the generated text. Notice that all average scores are above 3 except for English/S2 (2.98).

The remaining rows in the table provide additional details. We carried out a specific statistical analysis for each category. Regarding language, we applied the [Wilcoxon, 1945] non-parametric test as ranking test (with significance level $\alpha = 0.05$). The null hypothesis (H_0: "the medians of the differences between the two group samples are equal") was rejected (p-value=0.023). Regarding background, H_0 was not rejected (p-value=0.125) by the Wilcoxon test.

In the light of these results we observe significant statistical difference between scores given by Italian and native English speakers, but no difference was appreciated with respect to their BSc background. Moreover, the lower score provided (by subjects without BSc) for S1 and S3 may suggest that the current explanation is too technical.

Notice that all previous tests were carried out with the STAC web platform [Rodriguez-Fdez *et al.*, 2015].

4 Conclusions and Future Work

The proposed method is a first approach for hybridizing ML and KB systems in order to realize eXplainable Artificial Intelligence. Preliminary experimental results are encouraging since the resulting XAI systems show comparable accuracy with respect to classical ML methods, but they also provide an explanation of the inference process in natural language. This explanation has been subject to human assessment in our case study. Results show the feasibility of the approach, but there is room for further research.

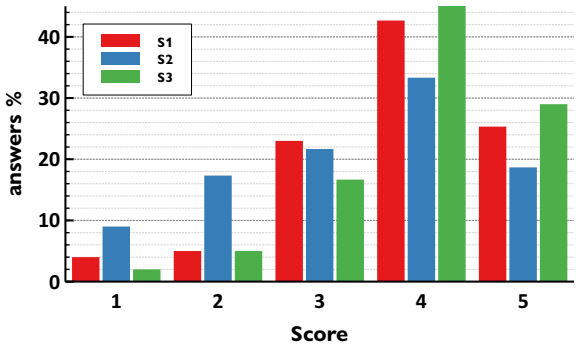


Figure 4: Histogram of the scores obtained in the survey.

The proposed XAI system is general enough for further customizations. It is worthy to note that we are not explaining how an opaque ML system works but building a whole XAI system with a good interpretability-accuracy trade-off; where the high accuracy is due to an opaque ML component while the high interpretability relies on the designed ML+KB explainable component. Thus, it is possible to use more sophisticated ML methods in order to achieve the best possible accuracy. Moreover, it is possible to use other interpretable ML and KB models, according to the application needs, provided that they can be represented in GLMP.

Future research is devoted in widening the representation possibilities of GLMP and, consequently, the overall NLG framework, with the aim of providing users with clearer and more actionable explanations. Moreover, we plan to enhance the hybrid XAI approach with further reasoning capabilities.

Acknowledgments

Supported by RYC-2016-19802 (Ramón y Cajal contract), and 2 MINECO projects TIN2017-84796-C2-1-R (BIGBISC) and TIN2014-56633-C3-3-R (ABS4SOW), funded by the Spanish "Ministerio de Economía y Competitividad". A. Ramos-Soto is funded by the "Consellería de Cultura, Educación e Ordenación Universitaria" (under the Postdoctoral Fellowship accreditation 481B 2017/030). Financial support from the Xunta de Galicia (Centro singular de investigación de Galicia accreditation 2016-2019) and the European Union (European Regional Development Fund - ERDF), is gratefully acknowledged.

References

[Alonso and Magdalena, 2011] J. M. Alonso and L. Magdalena. HILK++: An interpretability-guided fuzzy modeling methodology for learning readable and comprehensible fuzzy rule-based classifiers. *Soft Computing*, 15:1959–1980, 2011.

[Alonso *et al.*, 2015] J. M. Alonso, C. Castiello, and C. Mencar. Interpretability of fuzzy systems: Current research trends and prospects. In *Handbook of Computational Intelligence*, pages 219–237. Springer, 2015.

[Biran and Cotton, 2017] O. Biran and C. Cotton. Explanation and justification in machine learning: A survey. In *Proceedings of the IJCAI-17 Workshop on Explainable AI (XAI)*, pages 8–13, Melbourne, Australia, 2017.

[Breiman, 2001] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[Buchanan and Shortliffe, 1984] B. Buchanan and E. Shortliffe. *Rule-based expert systems: the MYCIN experiments of the Stanford Heuristic Programming Project*. Addison-Wesley, 1984.

[Castellano *et al.*, 2017] G. Castellano, C. Castiello, and A. M. Fanelli. The FISDeT software: Application to beer style classification. In *Proceedings of the IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 1–6, Naples, Italy, 2017.

[Conde-Clemente *et al.*, 2017] P. Conde-Clemente, J. M. Alonso, and G. Trivino. rLDCP: R package for text generation from data. In *Proceedings of the IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 1–6, Naples, Italy, 2017.

[Dale, 2017] R. Dale. The commercial NLP landscape in 2017. *Natural Language in Engineering*, 23(4):641–647, 2017.

[Gaines and Boose, 1988] B. R. Gaines and J. H. Boose. *Knowledge acquisition for knowledge-based systems.* Academic Press, 1988.

[Gatt and Krahmer, 2018] A. Gatt and E. Krahmer. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *Journal of Artificial Intelligence Research*, 61:65–170, 2018.

[Goodman and Flaxman, 2016] B. Goodman and S. Flaxman. European union regulations on algorithmic decision-making and a "right to explanation". In *ICML Workshop on Human Interpretability in Machine Learning (WHI)*, pages 1–9, New York, NY, 2016. https://arxiv.org/abs/1606.08813.

[Pancho *et al.*, 2013] D. P. Pancho, J. M. Alonso, and L. Magdalena. Quest for interpretability-accuracy tradeoff supported by fingrams into the fuzzy modeling tool GUAJE. *International Journal of Computational Intelligence Systems*, 6(sup1):46–60, 2013.

[Quinlan, 1993] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, San Mateo, CA, 1993.

[Ramos-Soto *et al.*, 2016] A. Ramos-Soto, A. Bugarin, and S. Barro. On the role of linguistic descriptions of data in the building of natural language generation systems. *Fuzzy Sets and Systems*, 285:31–51, 2016.

[Reiter and Dale, 2000] E. Reiter and R. Dale. *Building natural language generation systems*. Cambridge University Press, 2000.

[Rodriguez-Fdez *et al.*, 2015] I. Rodriguez-Fdez, A. Canosa, M. Mucientes, and A. Bugarin. STAC: A web platform for the comparison of algorithms using statistical tests. In *Proceedings of the IEEE international conference on fuzzy systems (FUZZ-IEEE)*, pages 1–8, Istanbul, Turkey, 2015.

[Rosenblatt, 1961] F. Rosenblatt. *Principles of neurodynamics: Perceptions and the theory of brain mechanism*. Spartan Books, Washington, DC, 1961.

[Tintarev and Masthoff, 2015] N. Tintarev and J. Masthoff. Explaining recommendations: Design and evaluation. In F. Ricci, L. Rokach, and B. Shapira, editors, *Recommender Systems Handbook*, pages 353–382. Springer, Boston, MA, 2015.

[Trivino and Sugeno, 2013] G. Trivino and M. Sugeno. Towards linguistic descriptions of phenomena. *International Journal of Approximate Reasoning*, 54:22–34, 2013.

[US Public Policy Council, 2017] ACM US Public Policy Council. Statement on Algorithmic Transparency and Accountability, 2017.

[Wachter *et al.*, 2017] S. Wachter, B. Mittelstadt, and L. Floridi. Why a right to explanation of automated decision-making does not exist in the general data protection regulation. *International Data Privacy Law*, 7(2):76–99, 2017.

[Wilcoxon, 1945] F. Wilcoxon. Individual comparisons by ranking methods. *Biometrics bulletin*, pages 80–83, 1945.

[Witten *et al.*, 2016] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 4th edition, 2016.

[Zadeh, 1965] Lotfi A. Zadeh. Fuzzy sets. *Information and Control*, 8(3):338–353, 1965.

[Zadeh, 1999] Lotfi A. Zadeh. From computing with numbers to computing with words - From manipulation of measurements to manipulation of perceptions. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 46(1):105–119, 1999.

[Zadeh, 2001] Lotfi A. Zadeh. A new direction in AI: Toward a computational theory of perceptions. *Artificial Intelligent Magazine*, 22(1):73–84, 2001.

Explanations for Temporal Recommendations

Homanga Bharadhwaj[1], **Shruti Joshi** [2]
Department of Computer Science and Engineering[1], Department of Electrical Engineering[2]
Indian Institute of Technology Kanpur, India[1,2]
homangab@iitk.ac.in[1], shrutij@iitk.ac.in[2]

Abstract

Recommendation systems are an integral part of Artificial Intelligence (AI) and have become increasingly important in the growing age of commercialization in AI. Deep learning (DL) techniques for recommendation systems (RS) provide powerful latent-feature models for effective recommendation but suffer from the major drawback of being non-interpretable. In this paper we describe a framework for explainable temporal recommendations in a DL model. We consider an LSTM based Recurrent Neural Network (RNN) architecture for recommendation and a neighbourhood-based scheme for generating explanations in the model. We demonstrate the effectiveness of our approach through experiments on the Netflix dataset by jointly optimizing for both prediction accuracy and explainability.

1 Introduction

Explainability in machine learning models has been a topic of intense research and debate. The issue of explainability in recommendation systems (RS) is all the more pertinent due to their sheer ubiquity. RS have become embedded in all forms of user-interface interaction and now form the core of our every day activities. So, to improve users' experience and trust, transparency and explainability are increasingly being incorporated in practical recommender systems. For instance, Netflix justifies the movies recommended by displaying similar movies obtained through the social network of users. Similarly, Amazon justifies its recommendations by showing similar items obtained through neighbourhood based Collaborative Filtering (CF). There has been growing research in Deep Learning (DL) based models for recommendations which are known for giving excellent prediction accuracies. Now there is almost a universal consensus regarding the fact that the main drawback of Deep Learning models is their non-interpretability, however there have been recent efforts towards mitigating this drawback such as [Binder *et al.*, 2016; Zhang and Wallace, 2015].

For RS, most explanation based methods either fall into the classical neighbourhood based Collaborative Filtering (CF) or rule-based methods [Pazzani and Billsus, 2007; Lops *et al.*, 2011]. CF based recommendation methods, which leverage the wisdom of the crowd are more popular due to their scalability and robustness. Some recent works such as [Abdollahi and Nasraoui, 2016a] and [Abdollahi and Nasraoui, 2016b] have integrated explanations into Matrix Factorization (which is a latent factor model), and into Restricted Boltzmann Machines respectively. These however cannot be applied to temporal recommendations which seek to model user preferences over time. Modeling temporal evolution of user preferences and item states for effective recommendation systems (RS) is an active area of research and recent publications have illustrated the effectiveness of Recurrent Neural Networks (RNN) [Wu *et al.*, 2017; Hidasi *et al.*, 2016] for the same.

Recurrent Recommender Networks [Wu *et al.*, 2017] is a powerful technique of temporal recommendations. We build our model on top of this architecture by incorporating a neighbourhood-style explanation scheme. Here, LSTM [Hochreiter and Schmidhuber, 1997] based RNNs are used for modelling user and item latent states. The specific domain of movie recommendation is targeted in the experiments but the method is fairly generalizable across domains. We first formalize the notion of explainability by defining a time-varying bipartite graph between users and items such that the edge weights measure a notion of explainability of an item for a user by exploiting ratings of other users similar to the one in question. To optimize for explainability in addition to the prediction accuracy, we include a term in the optimization objective that seeks to minimize the distance between latent features of items and users weighed by their explainability as defined previously.

2 Related Works

2.1 Explainable Recommendations

Explanations in recommendation systems has been a topic of active research for a very long time, motivated by the general consensus that modern RS algorithms have been *black boxes* offering no transparency or human-interpretable insights. Although the underlying algorithm of a recommendation framework may influence the type of explanations generated, it is also an ecologically valid method to have a completely separate engine for generating explanations [Tintarev and Masthoff, 2011]. This is particularly interesting for com-

plex RS models like those using collaborative filtering and/or deep learning techniques [Herlocker *et al.*, 2000; Hu *et al.*, 2008]. Some methods generate explanations based on auxiliary data like review texts [Zhang, 2015] while others do not require additional information (apart from that used in the recommendation algorithm) for generating explanations [Abdollahi and Nasraoui, 2016b; Abdollahi and Nasraoui, 2016a; Herlocker *et al.*, 2000].

2.2 Temporal Recommendations

The temporal evolution of user preferences and item states has been discussed in multiple previous works. Recent papers which have been very impactful include [Donkers *et al.*, 2017] and RRN [Wu *et al.*, 2017]. [Donkers *et al.*, 2017] developed a Gated Recurrent Unit(GRU) [Cho *et al.*, 2014] based RNN for sequential modelling of user preferences. A rectified linear integration of user states is done in their modified GRU cell to evolve each user state for personalized recommendation. On the other hand, RRN targeted movie recommendation tasks by exploiting explicit feedback from users. Customized LSTM cells were used for modelling the function through which user and item latent states evolve. [Devooght and Bersini, 2017] leverage sequence information to mitigate the scarcity of user-item interactions and identify items that will be eventually preferred and that those that will be immediately desired. Since our architecture is built over RRN, we elaborate on the details in Section 3.2.

3 The Model

3.1 Explanations

Our model uses only the users' movie rating data for predicting as well as explaining recommendations. This is in contrast to many previous studies that consider other data variables such as user demographics, text reviews [Zhang, 2015], user preferences for entities associated with the items to be recommended [He *et al.*, 2015], etc for predicting recommendations, as well as studies that use these data variables as auxiliary information for only explaining their recommendations [Herlocker *et al.*, 2000]. We employ a neighbourhood based explanation scheme which is similar to [Abdollahi and Nasraoui, 2016b] and formalize the definition of neighbourhood and explainability as follows.

Neighbourhood is calculated based on discounted cosine similarity between the users' (say user i and user k) as $sim_{i,k} = \sum_{t,m} \gamma_t r_{im|t} \cdot r_{km|t}$, where the discounting factor, $\gamma_t = \frac{1}{1+t}$ and $r_{im|t}$ represents user i's rating at time t for a movie m. We then pick the p most similar users as the neighbourhood of user i.

This notion of a neighbourhood style explanation has also been employed in various graph based models such as [He *et al.*, 2015]. We define a temporally varying bipartite graph between the set of users and the set of items with the following edge weight matrix

$$M_{umt} = \frac{\sum_{z \in Q_t^p(u)} r_{zm|t}}{p \times max\{r_{zm|t} \forall z \in Q_t^p(u)\}} \quad (1)$$

Here $Q_t^p(u)$ is the set of p neighbours for user u and $r_{zm|t}$ is the rating of user z of movie m at time step t. It is important to note that M_{umt}, which represents the edge-weight between user u and movie m at time step t is a real number between 0 and 1. It has the interpretation of being a quantification of how explainable is movie m for user u. $r_{zm|t}$ is 0 if user z has not rated item m at time step t. So, a value of $M_{umt} = 0$ would mean that none of the users in the neighborhood of user u have rated movie m at time step t and hence movie m is not explainable to user u at that time step.

In addition, since we postulate that there are stationary components as well in the latent features of users and items (Section 3.2) we also develop a stationary bipartite graph between the set of users and the set of items for explanations. This is in addition to the above time varying interpretation. For this bipartite graph, the edge weight matrix is described as

$$M_{um} = \frac{\sum_{z \in Q^p(u)} r_{zm}}{p \times max\{r_{zm} \forall z \in Q^p(u)\}} \quad (2)$$

Here, all the terms have the same meaning as in the previous equation, but they appear without the time index.

3.2 The Rating Prediction Model

We use LSTM based Recurrent Neural Networks for modelling the temporal evolution of user and movie latent states. The approach is essentially matrix factorization through time with the temporal evolution of latent factors being modelled by LSTM based Recurrent Neural Networks. This approach is similar in spirit to RRN [Wu *et al.*, 2017] and the novelty of our method is the incorporation of explainability which we describe in Section 3.3.

We denote by u_{it} and m_{jt}, the latent features of user i and movie j respectively at time index t. Let $\hat{r}_{ij|t}$ denote the predicted rating for user i on movie j at time index t and let $r_{ij|t}$ be the actual rating. $\{r_{ij|t}\}_j$ denotes the set of ratings for all movies j. We define the following model for updates

$$\hat{r}_{ij|t} = f(u_{it}, m_{jt})$$
$$u_{i,t+1} = g(u_{it}, \{r_{ij|t}\}_j)$$
$$m_{i,t+1} = h(m_{it}, \{r_{ij|t}\}_j)$$

The functions f, g, h are learned by the model to infer user and movie states. Section 3.1 of the RRN paper [Wu *et al.*, 2017] elaborates on the user and movie state formulation, which we mention briefly below.

$$y_t = V[x_t, 1, \tau_t, \tau_{t-1}], u_t = LSTM(u_{t-1}, y_t)$$

Here τ_t represents the wallclock at time t, x_t is the rating vector for the user and V represents the transformation. Similar to the approach followed in RRN [Wu *et al.*, 2017], we include the profile identities of the user and the movie as the stationary components of the rating in addition to their time varying state vectors. So,

$$\hat{r}_{ij|t} = f(u_{it}, m_{jt}, u_i, m_j)$$
$$= \langle \tilde{u}_{it}, \tilde{m}_{jt} \rangle + \langle u_i, m_j \rangle$$

where $\tilde{u}_{it} = Au_{it} + c$ & $\tilde{m}_{jt} = Bm_{jt} + d$. This decomposition makes it evident how RRN is essentially matrix factorization through time as mentioned earlier.

Figure 1: Shown above are two instances of top-3 recommendations, decreasing left to right in confidence. Here, Epoch 1: within a month before present, Epoch 2: within one year before present, and Epoch 3: $\geq$ 1 year before present. In instance a) dominance of recency in ranking is seen, while in b) the dominance of the sum of the ranking is evident. These are anecdotal examples from the evaluation of TemEx-Fluid on the Netflix dataset.

3.3 Incorporating Explainability in the Model

Since the user and item (movie) states are time varying, we need a time varying bipartite graph which is defined by a time varying edge-weight matrix M_{ijt}. If movie j is explainable to user i in at time step t, then their latent representations $\tilde{m}_{jt}$ and $\tilde{u}_{it}$ respectively must be close. Based on this intuition, we include the term $(\tilde{m}_{jt} - \tilde{u}_{it})^2 M_{ijt}$ in our minimization objective. We formulate the overall objective in such a way that both prediction accuracy as well as explainability are optimized. So, if there are two movies which are likely to be equally preferred by the user, the model will recommend the movie which is more explainable. It is important to note that explainability and prediction accuracy may be at odds for some user-movie pairs and hence we need to define a *joint* objective function including both the aspects, which is defined as follows

$$L = \sum_{i,j}(\sum_{t}(r_{ij|t} - \hat{r}_{ij|t}(\theta))^2 + \alpha(\tilde{m}_{jt} - \tilde{u}_{it})^2 M_{ijt}) + \beta(m_{jt} - u_{it})^2 M_{ij}) + R(\theta)$$

The benefit of having temporally varying explanation-graphs is that the generated explanations aren't the conventional '7/10 people with similar interests as you have rated this movie 4 and higher" but can employ information related to rating distribution across time too, as seen in Figure 1.

If we use the heuristic that explanations in the near past are more *relevant* than those in the far past, we can weigh the term $(\tilde{m}_{jt} - \tilde{u}_{it})^2 M_{ijt}$ for explanations by a temporally decaying factor $\alpha(t)$. In this paper, we use the specific form of $\alpha(t)$ to be $\exp(-\alpha t)$ but there are other popular choices of this discount factor as well [cite other decaying functions used in ML training]. So, the modified objective function becomes

$$L' = \sum_{i,j}(\sum_{t}(r_{ij|t} - \hat{r}_{ij|t}(\theta))^2 + e^{(-\alpha t)}(\tilde{m}_{jt} - \tilde{u}_{it})^2 M_{ijt}) + \beta(m_{jt} - u_{it})^2 M_{ij}) + R(\theta)$$

Here, we call the model corresponding to the objective function L as TempEx-Dry and the one corresponding to L' as TempEx-Fluid. We then perform simulations for both the objective functions and compare their pros and cons empirically.

3.4 Training

Although the conventional method of training Recurrent Neural Networks is Backpropagation Through Time (BPTT), as mentioned in [Wu *et al.*, 2017], backpropagation through two sequences (rating depends on both user state RNN and item state RNN) is computationally infeasible. We adopt the strategy of subspace descent as done in [Wu *et al.*, 2017] to alleviate this problem. In practice, we found that using Dropout [Srivastava *et al.*, 2014] helps in stabilizing the gradients and preventing over-fitting due to the additional terms introduced in the minimization objective. The hyperparameters α and β were tuned by a grid-search in the range 0 to 1 and the tuned value of α is kept at 0.4, while that of β is kept at 0.6 throughout all experiments.

4 Simulation Studies

4.1 Setup

Through a series of simulation experiments, we seek to understand two basic questions, 1) How effective is the model in generating explanations? and 2) What is the trade-off between prediction accuracy and explainability? All our experiments have been done using Tensorflow r1.4 [Abadi *et al.*, 2016] in Python 3. We use ADAM optimizer during training [Kingma and Ba, 2014]. We perform all experiments on a timestamped Netflix dataset used in [Wu *et al.*, 2017], which was first used in [Diao *et al.*, 2014]. It consists of 100M movie ratings from 1999 to 2005, where each data point is a (user-id, item-id, time-stamp, rating) tuple with a time stamp granularity of 1 day. For consistency, we use the same preprocessing and train-test split as in [Wu *et al.*, 2017].

4.2 Benefit of Explanations

To answer 1), we use standard IR metrics like precision and recall with the notion of explainability. [Abdollahi and Nasraoui, 2016b] introduces Mean Explainable Precision (MEP) and Mean Explainable Recall (MER) metrics. To state briefly, MEP is defined as the ratio of the number of explainable items recommended to the total recommended items for each user averaged over all users. Similarly, MER is the number of explainable items recommended to the total number of explainable items for each user averaged over all users. We benchmark our performance against state of the art models such

Table 1: RMSE and MRR for benchmark models at $p = 50$ for the Netflix dataset

	RRN	T-SVD	PMF	ERBM	EMF	TempEx-Dry	TempEx-Fluid
MRR	0.371	0.342	0.322	0.321	0.318	0.374	0.382
RMSE	0.922	0.927	0.925	0.931	0.934	0.923	0.919

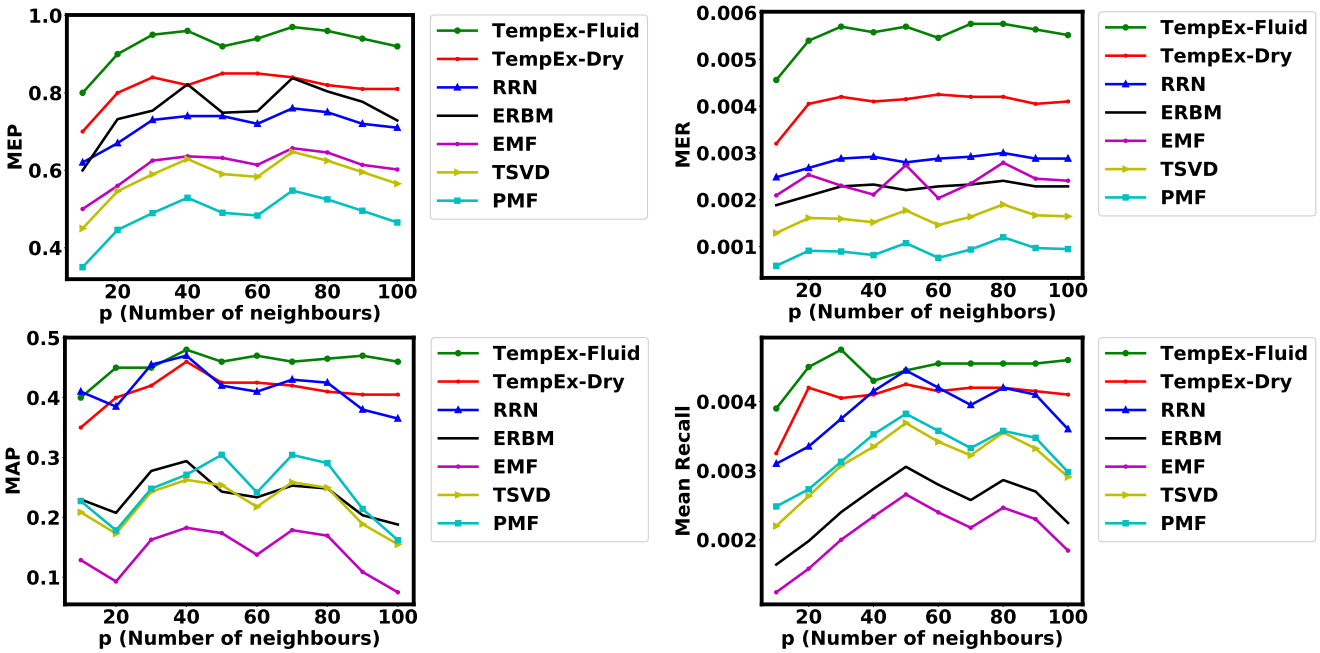


Figure 2: MEP, MER, Mean Average Precision (MAP) and MR (Mean Recall) for benchmark models with varying number of neighbours. All the values are averaged over the test set for all users

as RRN, T-SVD, PMF and recent explainable CF methods like EMF and ERBM [Wu *et al.*, 2017; Mnih and Salakhutdinov, 2008; Koren, 2010; Abdollahi and Nasraoui, 2016a; Abdollahi and Nasraoui, 2016b]. We also evaluate two versions of our model- without incorporation of the temporally weighted explanation term (TempEx-Dry), and with an exponentially decaying temporal weight on explanations (TempEx-Fluid).

As revealed in the top row of Figure 2, TempEx-Fluid has the best measure of explainability across all values of p and it also performs consistently better than TempEx-Dry. So, temporally weighing the explainability term indeed leads to a more explainable model.

4.3 Tradeoff between Explanations and Prediction Accuracy

To answer 2), we evaluate the performance of our model against benchmark models on standard metrics like RMSE, Mean Average Precision (MAP), Mean Recall (MR) and Mean Reciprocal Rank (MRR). The purpose of this evaluation is to see whether incorporation of the explainability terms in the optimization objective leads to substantial gains /losses on the actual prediction accuracy of ratings.

Table 1 and the bottom row of Figure 2 shows the results of our analysis. At $p = 50$ we observe that the RMSE and MRR values of TempEx-Fluid are higher than the standard RRN model. This indicates that incorporating explainability also improves the prediction accuracy on the test set by imposing an additional regularizer in the model. The bottom row of Figure 2 shows that TempEx-Fluid consistently has better performance on Mean Average Precision and Mean Recall for all values of p. This leads further credence to the fact that there is no compromise being made on prediction accuracy by including explainability. Also, a comparison between TempEx-Dry and TempEx-Fluid reveals that TempEx-Fluid performs consistently better for all values of p. This is due to the more nuanced temporal decay in the optimization objective which appropriately weighs down past effects.

5 Conclusion

In this paper, we devised a methodology of incorporating explanations in time-series recommendation. We devised a time-varying neighbourhood style explanation scheme and jointly optimized for prediction accuracy and explainability. Through simulation results we demonstrated the efficacy of the proposed framework. It is important to note that our method of explanation is different from the core recommendation algorithm, which is a common practice in explainable RS. However, as future work we plan to devise an explanation scheme that tries to explain the recommendation algorithm. Since, our algorithm is a deep learning model, we need to incorporate schemes like layer-wise relevance propagation that seek to propagate the relevance of the output through the layers of the network and assign relevance to the inputs.

References

[Abadi *et al.*, 2016] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283, 2016.

[Abdollahi and Nasraoui, 2016a] Behnoush Abdollahi and Olfa Nasraoui. Explainable matrix factorization for collaborative filtering. In *Proceedings of the 25th International Conference Companion on World Wide Web*, pages 5–6. International World Wide Web Conferences Steering Committee, 2016.

[Abdollahi and Nasraoui, 2016b] Behnoush Abdollahi and Olfa Nasraoui. Explainable restricted boltzmann machines for collaborative filtering. *arXiv preprint arXiv:1606.07129*, 2016.

[Binder *et al.*, 2016] Alexander Binder, Sebastian Bach, Gregoire Montavon, Klaus-Robert Müller, and Wojciech Samek. Layer-wise relevance propagation for deep neural network architectures. In *Information Science and Applications (ICISA) 2016*, pages 913–922. Springer, 2016.

[Cho *et al.*, 2014] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

[Devooght and Bersini, 2017] Robin Devooght and Hugues Bersini. Long and short-term recommendations with recurrent neural networks. In *Proceedings of the 25th Conference on User Modeling, Adaptation and Personalization*, pages 13–21. ACM, 2017.

[Diao *et al.*, 2014] Qiming Diao, Minghui Qiu, Chao-Yuan Wu, Alexander J Smola, Jing Jiang, and Chong Wang. Jointly modeling aspects, ratings and sentiments for movie recommendation (jmars). In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 193–202. ACM, 2014.

[Donkers *et al.*, 2017] Tim Donkers, Benedikt Loepp, and Jürgen Ziegler. Sequential user-based recurrent neural network recommendations. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*, pages 152–160. ACM, 2017.

[He *et al.*, 2015] Xiangnan He, Tao Chen, Min-Yen Kan, and Xiao Chen. Trirank: Review-aware explainable recommendation by modeling aspects. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 1661–1670. ACM, 2015.

[Herlocker *et al.*, 2000] Jonathan L Herlocker, Joseph A Konstan, and John Riedl. Explaining collaborative filtering recommendations. In *Proceedings of the 2000 ACM conference on Computer supported cooperative work*, pages 241–250. ACM, 2000.

[Hidasi *et al.*, 2016] Balázs Hidasi, Massimo Quadrana, Alexandros Karatzoglou, and Domonkos Tikk. Parallel recurrent neural network architectures for feature-rich session-based recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 241–248. ACM, 2016.

[Hochreiter and Schmidhuber, 1997] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[Hu *et al.*, 2008] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pages 263–272. Ieee, 2008.

[Kingma and Ba, 2014] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[Koren, 2010] Yehuda Koren. Collaborative filtering with temporal dynamics. *Communications of the ACM*, 53(4):89–97, 2010.

[Lops *et al.*, 2011] Pasquale Lops, Marco De Gemmis, and Giovanni Semeraro. Content-based recommender systems: State of the art and trends. In *Recommender systems handbook*, pages 73–105. Springer, 2011.

[Mnih and Salakhutdinov, 2008] Andriy Mnih and Ruslan R Salakhutdinov. Probabilistic matrix factorization. In *Advances in neural information processing systems*, pages 1257–1264, 2008.

[Pazzani and Billsus, 2007] Michael J Pazzani and Daniel Billsus. Content-based recommendation systems. In *The adaptive web*, pages 325–341. Springer, 2007.

[Srivastava *et al.*, 2014] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[Tintarev and Masthoff, 2011] Nava Tintarev and Judith Masthoff. Designing and evaluating explanations for recommender systems. In *Recommender systems handbook*, pages 479–510. Springer, 2011.

[Wu *et al.*, 2017] Chao-Yuan Wu, Amr Ahmed, Alex Beutel, Alexander J Smola, and How Jing. Recurrent recommender networks. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pages 495–503. ACM, 2017.

[Zhang and Wallace, 2015] Ye Zhang and Byron Wallace. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *arXiv preprint arXiv:1510.03820*, 2015.

[Zhang, 2015] Yongfeng Zhang. Incorporating phrase-level sentiment analysis on textual reviews for personalized recommendation. In *Proceedings of the eighth ACM international conference on web search and data mining*, pages 435–440. ACM, 2015.

Towards Providing Explanations for AI Planner Decisions

Rita Borgo, Michael Cashmore, Daniele Magazzeni
King's College London
firstname.lastname@kcl.ac.uk

Abstract

In order to engender trust in AI, humans must understand what an AI system is trying to achieve, and why. To overcome this problem, the underlying AI process must produce justifications and explanations that are both transparent and comprehensible to the user. AI Planning is well placed to be able to address this challenge.

In this paper we present a methodology to provide initial explanations for the decisions made by the planner. Explanations are created by allowing the user to suggest alternative actions in plans and then compare the resulting plans with the one found by the planner. The methodology is implemented in the new XAI-PLAN framework.

1 Introduction

Artificial Intelligence technologies are increasingly ubiquitous in modern society and have the potential to fundamentally change all aspects of our lives. At the same time, increasing concerns are being raised about the transparency and accountability of AI systems.

The European Parliament resolved in 2017 to require AI systems to follow a principle of transparency, meaning they should be able to justify their decisions in a manner comprehensible to a human. Similarly, legal measures are being adopted to provide individuals affected by automated decision-making with a "right to explanation", as referred to in the recent EU General Data Protection Regulation (GDPR), in place from May 2018.

In order to engender trust in AI, humans must understand what an AI system is trying to achieve, and why. To overcome this problem, the underlying AI process must produce justifications and explanations that are both transparent and comprehensible to the user.

In 2016 DARPA launched the Explainable AI program, and since then the AI research community has been looking at this challenge with growing interest[1], mainly trying to explain or verify neural networks. While much current attention is focussed on the recent advancements in data-driven AI (e.g., Machine Learning, Deep Learning), model-based AI such as Planning is well placed to be able to address the challenges of transparency and explainability.

In AI Planning, most previous work within the realm of explainability aims to help humans understand the elements of a plan suggested by the system (e.g. [Sohrabi *et al.*, 2011],). This involves the transformation of planner output into forms that non-expert users can understand and the description of causal and temporal relations between plan steps. However, describing how a single plan works is a different problem than explaining why the planner suggests a particular plan. This requires new fundamental algorithms for generating explanations. Now, planning is being used in new critical domains (e.g., smart grid [Thiébaux *et al.*, 2013; Piacentini *et al.*, 2016], urban/air traffic control [Vallati *et al.*, 2016; Morris *et al.*, 2015], mining [Lipovetzky *et al.*, 2014], underwater robotics [Cashmore *et al.*, 2018]) and handles temporal constraints, numeric resources and continuous change [Piotrowski *et al.*, 2016], resulting in more complex plans. In this regard, explanation-generating algorithms will be instrumental in developing more robust systems for use in these critical domains.

In [Fox *et al.*, 2017] a roadmap for addressing *Explainable Planning* is proposed, which identifies a list of questions that planning systems should be able to answer, both offline before the plan is approved as well as online during plan execution. In this paper we tackle some of these questions, and in particular we focus on what we think are the most common questions a user would ask when confronted with a plan: "*Why would you do this rather than that?*", and "*Why is what you suggest more efficient than something else?*". Confronted with a question or an alternative indicated by a human user, the explanation should be a demonstration that the alternative would prevent the generation of a valid plan, or at least be no better than the existing plan. This would be a justification for the choices made by the planner. An important side effect of such an approach is that this interaction between the user and the planner enhances mixed-initiative planning, and it might be the case that the suggestion made by the user actually improves the final plan. When times and real numbers are taken into account, as it is the case in many real-world scenarios, one cannot expect optimal plans from a planner, and hence the knowledge of the domain expert should be considered in the planning process.

In this paper we present a methodology to provide expla-

[1] See Workshops on Explainable AI at IJCAI-17 and IJCAI-18.

nation for the decisions made by the planner. Explanations are created by allowing the user to explore alternative actions in the plans and then compare the resulting plans with the one found by the planner, using different metrics. We implemented the methodology in the new XAI-PLAN framework. The methodology is domain-independent and is agnostic about the planning system used. We evaluated the framework in a number of domains using ROSPlan [Cashmore *et al.*, 2015].

The paper is structured as follows. In Section 2 we provide a brief overview of related work. In Section 3 we present the methodology and in Section 4 we demonstrate the methodology with a working example. In Section 5 we describe the implementation of the XAI-PLAN framework, and Section 6 concludes the paper.

2 Related Work

Plan Explanation is an area of planning where the main goal is to help humans understand the plans produced by the planners (e.g., [Sohrabi *et al.*, 2011; Seegebarth *et al.*, 2012; Bidot *et al.*, 2010]). This involves the translation of plans to forms that humans can easily understand and the design of interfaces that help this understanding. Relevant works in robotics include [Hayes and Shah, 2017; Rosenthal *et al.*, 2016]. Similar work focuses on generating diverse solutions when user preferences are not known [Nguyen *et al.*, 2012], or top-k solutions [Katz *et al.*, 2018]. While providing the user with more choice, this does not necessarily provide explanation, nor prevent the user asking why a particular plan is selected.

Kambhampati and his team focus on the important scenario where humans and the agent have different models of the world. Explanations in that context must handle the issue of *model reconciliation* [Chakraborti *et al.*, 2017; Sreedharan *et al.*, 2018]. In the same context, *Plan Explicability* [Zhang *et al.*, 2017] focuses on human interpretation of plans. In this stream of works, the focus is on optimal plans in classical planning, which might differ because of the different models used to generate them. We focus on more expressive domains where the model is well defined, but the resulting state space is too vast and complex. In cases where the model is sufficiently complex, it is not possible to provide explanations that can be well understood in the form of model reconciliation. In this line of research, relevant works include [Smith, 2012; Langley *et al.*, 2017; Fox *et al.*, 2017].

3 Explanations for Planner Decisions

When confronted with a plan generated by a planner, a common question the user might ask is: "*Why does the plan contain this action rather than this other action that I would expect?*". Indeed, it should be noted that the support of AI is even more relevant when the AI suggests to do something *different* from what the user would do (and in particular a domain expert). At the same time such a different action plan would need an explanation before the user can be confident on its effectiveness and approve the plan.

For an effective explanation, rewriting the steps of the planning algorithm in natural language is not what is required. Nor is it very helpful to provide the heuristic evaluations of the states selected by the planner when searching for a plan [Fox *et al.*, 2017].

Rather, we argue that what can justify the selection of a set of actions by a planner is that this set of actions proves to be better, or at least no worse, than the set of actions the user would select. To this aim, a framework for providing explanations should allow the user to explore alternative actions in the plan and then compare the resulting plans with the one found by the planner. Different metrics can be used to evaluate the quality of different plans. Such an approach would increase the confidence of the user in the planner and would give him/her evidence for accepting or rejecting plans.

In this work, we use this approach to tackle the first three questions considered in the roadmap for explainable planning proposed in [Fox *et al.*, 2017]:

- Q1 Why did you do that?
- Q2 Why didn't you do something else (that I would have done)?
- Q3 Why is what you propose to do more efficient than something else (that I would have done)?

In order to evaluate different alternatives, it is necessary to infer by what metric the alternatives are to be compared (one plan might be longer but cheaper than a second — depending on the relative values of time and money, either plan might be considered better). Furthermore, the user might want to change more than one action in the current plan, or iteratively revise the plan, or explore more than one alternative for a given action. This is represented by the diagram in Figure 1 proposed in [Fox *et al.*, 2017]. Finally, as the set of possible alternatives the user might consider is potentially infinite, it is necessary to drive the user in the questions he/she can ask.

We considered all these issues in the development of XAI-PLAN, which is a methodology for providing explanations according to the view described above, and also gives the name to the framework implementing such methodology.

3.1 The XAI-PLAN Methodology

XAI-PLAN is based on the idea that the user should be allowed to explore alternative plans by suggesting different actions in the plan. This paradigm provides an answer to questions like,

Why does the plan contain action A rather than action B (that I would expect)?

The planning system is then used not only to generate the initial plan, but also to explore the alternative plans resulting from the user suggestions.

More formally, the XAI-PLAN methodology is presented in Algorithm 1. This algorithm takes as input an initial set of plans, which at the beginning contains only a single plan. The user selects an action in the plan (line 2), and this corresponds to *action A* in the question template above. As said before, the part of the question *"rather than action B"* is open to a possibly infinite set of instances. To this end, XAI-PLAN restricts this set to the applicable actions (line 3). The list of applicable actions is generated in the following way: first the state in which action a is applied in plan π is obtained. Then,

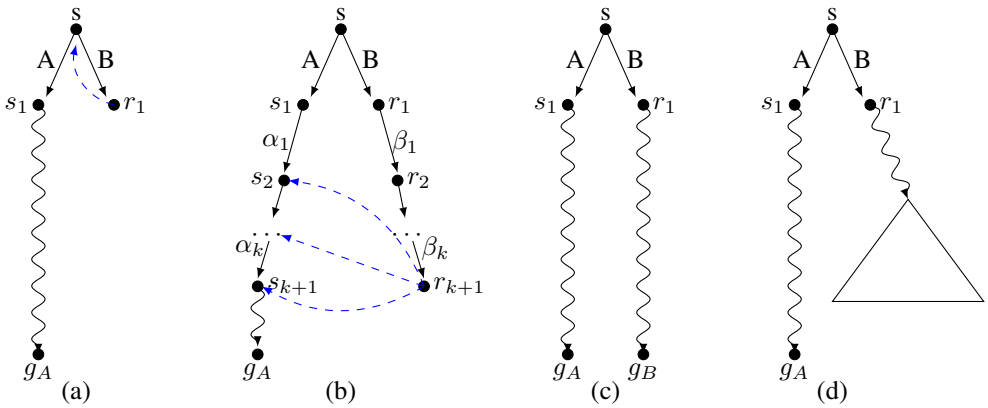


Figure 1: Possible plan behaviours after human-decision injection.

the set of all ground actions are filtered to only those whose preconditions are achieved in that state, minus the original action a. These actions are presented to the user, who can then select one of them (line 4).

Algorithm 1: XAI-PLAN

Input: initial set of plans Π
Loop
  ▷ User chooses an existing plan
1 : $\pi \leftarrow selectPlan(\Pi)$
  ▷ User chooses an action within that plan
2 : $a \leftarrow selectAction(\pi)$
  ▷ Generate list of alternative actions
3 : $applicable\_actions \leftarrow generateActions(\pi, a)$
  ▷ User selects an alternative action
4 : $a' \leftarrow selectAction(applicable\_actions)$
  ▷ New plan is generated
5 : $\pi' \leftarrow generatePlan(\pi, a, a')$
6 : $\Pi \leftarrow \Pi \cup \pi'$
EndLoop

Given a suggested action, a new plan is generated to answer the user's query (line 5). This can be done in one of four ways (described more in detail in the next section): planning from the initial state and forcing the user action to be performed, forcing the user action to be performed within a time-window, planning from the state after applying the user action, or planning both plan segments before and after the user action separately.

Finally, the new plan is added to the list of plans Π, which can be compared, and selected for further modifications (line 6). This allows iterative exploration of alternative plans.

3.2 Exploring Alternative Plans

After the user selects an alternative action, one way to explore alternative plans is to inject the user action in the plan and then replan from there. Figure 1 shows the possible outcomes:

(a) One possible behaviour is that the planner simply undoes the effect of the user action in order to return as quickly as possible to the original plan. While this might be the most efficient solution, it is undesirable from the standpoint of plan explanation, as it does not show clearly if an alternative exists, and the comparative quality of that alternative plan.

(b) The reversal of the user action can be avoided by enforcing the planner does not revisit state s. The second behaviour illustrates that the new plan, through actions $B, \beta_1, \ldots, \beta_k$ does return to the original plan, by a more or less efficient route than actions $A, \alpha_1, \ldots, \alpha_k$.

(c) The third behaviour shows the case where a new plan is found by the planner, without returning to the original planned actions.

(d) The fourth behaviour shows the case where no new plan is discovered, as the planner is unable to return to the original plan, and no alternative path to the goal exists.

Note, however, that replanning after applying the user action is not the only option, as replanning from the initial state with additional constraints is also possible. In the XAI-PLAN framework, there are four implementations of $generatePlan$ in Algorithm 1. These are:

1. planning from the state after applying the user action,
2. planning from the initial state and forcing the user action to be performed,
3. planning from the initial state and forcing the user action to be performed within a time-window,
4. or planning both plan segments before and after the user action separately.

Planning from the state obtained after applying the user action is done by disallowing the undo action, and then replanning. Planning from the initial state is achieved by updating the domain model to include a new predicate, `(applied-user-action)`, which is included as an effect of a new operator `(user-action)`. The new fact achieved by the user action is added as a goal of the problem, ensuring that the action is applied at least once in the plan.

When planning again from the initial state, there is a risk that the user action is performed differently from what was intended by the user. For example, the action might be appended to the end of a complete plan, simply to

achieve the `applied-user-action` effect. In a temporal plan, the user is able to specify a time-window in which they would like the action to be performed. This is done by adding a new predicate to the domain that is a condition of the user action operator, and can be enabled and disabled by timed-initial-lieterals (TILs). For example, the fact `(applicable-user-action)` is added as a new start condition of the operator `(user-action)`. Then, two TILs are added to the problem,

`(at LB (applicable-user-action))` and
`(at UB (not (applicable-user-action)))`,

where LB is the lower bound on the time-window, and UB is the upper bound. This ensures that the action is applied in the time-window that interests the user.

The final strategy is to plan separately from the user action to the goal (the *later plan*) and from the initial state to the user action (*initial plan*). The final plan shown to the user is obtained by concatenating the initial plan, the user action, and the later plan. Planning the later plan is performed by planning from the state after applying the user action, using the original goals. Then, a new problem is generated with the same initial state as the original problem, and goal to achieve the weakest conditions of the later plan. The weakest conditions are those facts which are conditions for actions in the later plan, not already supported by effects. An example of each approach to plan generation is described in the next section.

4 Examples of Exploring Alternative Plans

In this section we provide some examples of the four approaches to generating a new plan using the user suggested action, and some discussion on the strengths and drawbacks of each option.

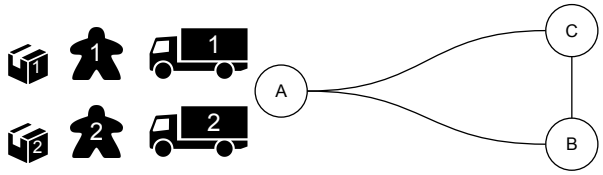


Figure 2: Problem setup for the DriverLog domain. The drivers, trucks, and packages are at location A.

Consider the problem shown in figure 2 from the *Driverlog* domain. In this problem two packages must be delivered to two separate locations. There are two drivers and two trucks available. Let us assume the plan found for this problem is shown in figure 3.

```
 0.0: (board-truck d2 t1 a) [10.0]
 0.0: (board-truck d1 t2 a) [10.0]
10.0: (load-truck p2 t1 a)  [10.0]
 10.0: (load-truck p1 t2 a) [10.0]
20.0: (drive-truck t2 a b d1) [30.0]
20.0: (drive-truck t1 a c d2) [30.0]
52.0: (unload-truck p1 t2 b)  [10.0]
52.0: (unload-truck p2 t1 c)  [10.0]
```
Figure 3: Plan generated for the driverlog problem. The highlighted action is selected by the user to be changed.

We explore the case in which the user wishes to see plans that only use one truck to deliver the package, so that they might keep one truck in reserve.

Example 1: Planning from the initial state

The user selects the highlighted action in figure 3 and chooses the alternative action, `(load-truck p1 t1 a)`, which loads the package into the first truck with the other package. A new plan must be generated from the initial state that achieves the goal and includes the action specified by the user.

This is done by updating the domain model to include a new predicate, `(applied-load-truck ?p ?t ?l)`, which is included as an effect of a new operator `(user-action-load-truck)`. The grounded fact `(applied-load-truck p1 t1 a)`, achieved by the user selected action, is added as a goal of the problem. In general, the new operator included in the domain is an exact copy of the operator selected by the user, with the addition of the `user-action` effect. The plan generated for this problem is shown in figure 4.

```
 0.0: (board-truck d2 t1 a) [10.0]
 0.0: (board-truck d1 t2 a) [10.0]
10.0: (load-truck p2 t2 a)  [10.0]
 10.0: (load-truck p1 t1 a) [10.0]
20.0: (drive-truck t2 a c d1) [30.0]
20.0: (drive-truck t1 a b d2) [30.0]
52.0: (unload-truck p1 t1 b)  [10.0]
52.0: (unload-truck p2 t2 c)  [10.0]
```
Figure 4: New plan generated from the initial state, forcing the highlighted user action.

When a time-window is specified, the user suggested `load-truck` action must be applied within a time-window, as descibed in section 3.1. The resulting plan for this problem is the same as that shown in figure 4. As we can see, the planner swaps which package to load in which truck.

The drawback of planning from the initial state is clear from this example – that the new plan still uses both trucks to deliver the packages. The question from the user, "why not use only one truck to deliver both packages?", is not given only by an action to be applied, but also the context in which that action should be applied. In this case, the action to load the package into truck `t1` in the state in which the other package was already loaded into `t1`.

Example 2: Planning after the user action

Planning from the state after applying the user action provides a plan that does show one truck delivering both packages, as shown in figure 5

```
 0.0: (board-truck d2 t1 a) [10.0]
 0.0: (board-truck d1 t2 a) [10.0]
10.0: (load-truck p2 t1 a)  [10.0]
 10.0: (load-truck p1 t1 a) [10.0]
20.0: (drive-truck t1 a b d2) [30.0]
52.0: (unload-truck p1 t1 b) [10.0]
62.0: (drive-truck t1 b c d2) [30.0]
94.0: (unload-truck p2 t1 c) [10.0]
```
Figure 5: Plan generated from the user action, using only one truck.

Consider the problem as shown in figure 6. This problem resembles the first with the addition that one driver is separated from the trucks by a long path. A plan generated from the state after the user action is shown in figure 7. The second driver, although no longer required, is still planned to walk along the long path to reach the trucks. This has the effect of dramatically reducing the plan's quality, and illustrates the drawback to planning from the user action – the quality of the plan shown to the user might be far from what is realistic. Although the plan generation is answering the question the user asks by considering the suggested action and state in which it was suggested, the answering plan is not a good answer.

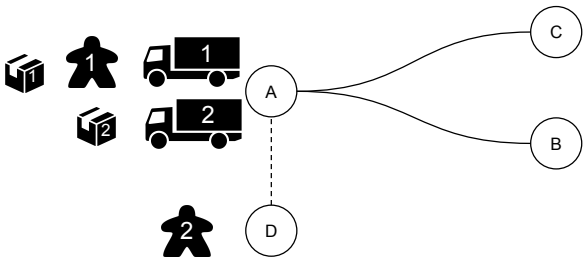


Figure 6: Problem setup for the DriverLog domain. The drivers, trucks, and packages are at location A.

```
 0.0: (walk d2 d a) [60.0]
62.0: (board-truck d2 t1 a) [10.0]
62.0: (board-truck d1 t2 a) [10.0]
72.0: (load-truck p2 t1 a)  [10.0]
72.0: (load-truck p1 t1 a) [10.0]
82.0: (drive-truck t1 a b d2) [30.0]
114.0: (unload-truck p1 t1 b) [10.0]
144.0: (drive-truck t1 b c d2) [30.0]
176.0: (unload-truck p2 t1 c) [10.0]
```
Figure 7: Plan generated from the user action, using only one truck. The second driver spends a long time walking.

Example 3: Planning before and after the user action
When planning the initial and later plans separately, the state in which the user suggested the action is not lost. The resultant plan, obtained by concatenating the initial plan, user action, and later plan is shown in figure 8. This plan uses only one truck, as the later plan was generated from the state of the user's suggested action, and also does not waste time waiting for the second driver, as the initial plan was also regenerated.

```
 0.0: (board-truck d1 t1 a) [10.0]
10.0: (load-truck p2 t1 a)  [10.0]
10.0: (load-truck p1 t1 a) [10.0]
20.0: (drive-truck t1 a b d1) [30.0]
52.0: (unload-truck p1 t1 b) [10.0]
62.0: (drive-truck t1 b c d1) [30.0]
94.0: (unload-truck p2 t1 c) [10.0]
```
Figure 8: Plan generated from the user action, using only one truck, and without waiting for the second driver.

However, this approach also has the drawback that the context of the user action might be lost. For example, if the user suggested that the first load action be altered, then the action is suggested in the context of the later plan, and not the state in which the action is suggested. Identifying exactly what context matters with respect to an action suggestion is a challenging topic that we will explore in future work.

5 The XAI-PLAN Framework

XAI-PLAN has been implemented using the ROSPlan framework [Cashmore *et al.*, 2015], which provides an interface to AI planning systems and a method for storing and modifying PDDL models in the Robot Operating System (ROS).

Figure 9 shows an overview of the XAI-PLAN framework. The XAI-Plan node implements the algorithms for generating explanatory plans, described in sections 3.1 and 4, and communicates to the user through a user interface described below. The *knowledge base*, *problem interface*, and *planner interface* are supplied by ROSPlan, which are used to store a PDDL model and provide an interface to the AI planner.

To provide a list of applicable actions to the user, the interface of the ROSPlan knowledge base is used to find the state at the point in the plan selected by the user, and then to retrieve the list of all grounded actions whose preconditions are achieved in that state.

Given an action suggested by a user, the domain is modified using the knowledge base interface, generating new predicates and operators that are required for the method of planning, as described in section 4. The initial state and goal of each problem instance are also set using this interface. The *problem interface* and *planning interface* nodes are then called to generate new plans for these problem instances.

The new plans are concatenated, if required and as described in section 4, and then shown to the user alongside a comparison with previously generated plans. The user controls these actions through a graphical user interface.

5.1 Interface Design

The interface was designed following a user-centered approach; each component either supports a user initiated action or caters for a possible user need, such as performance comparison or plan generation tracking. A screenshot of the interface is provided in Figure 10. The interface structure naturally unfolds the refinement cycles detailed in Algorithm 1.

A top bar (1) allows the user to select domain file, problem file, and planner. The *Plan* button (3), when pressed, initiates the plan generation process using the selected planner. The *Current Plan* section (2) holds the current plan, which initially will be the one generated by the planner without user intervention. Each action within the *Current Plan* window can be individually selected by the user. After selection, XAI-PLAN will prompt the planner to suggest feasible alternative actions to the user. The list of alternative actions is displayed inside the *Alternative Actions* section (4). The user is then allowed to either select any of the proposed alternative actions, or continue with the current plan and select a different action to compute a different set of alternative actions. Selection of any action within the *Alternative Actions* list will prompt the planner to generate an alternative plan which is displayed inside the *Alternative Plan* section (5).

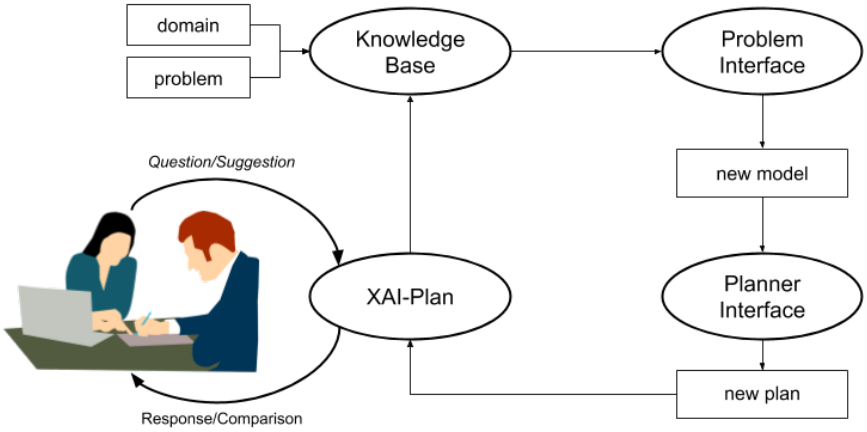


Figure 9: XAI-PLAN architecture and its integration with ROSPlan.

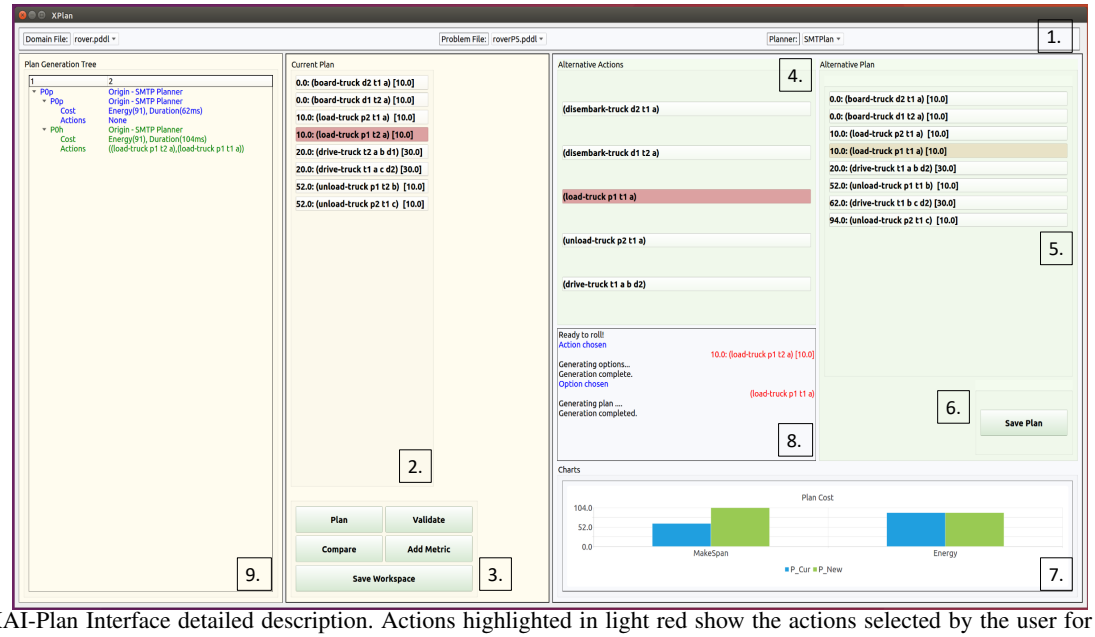


Figure 10: XAI-Plan Interface detailed description. Actions highlighted in light red show the actions selected by the user for replacement. The action highlighted in light green shows the selected alternative action within a newly generated plan.

A textbox (8) provides real-time feedback to the user, the textbox also allows user annotation to be added to the communication stream. Generated plans can be compared among each other via the *Compare* button (3). The user is allowed to define metrics to evaluate the quality of a plan via the *Add Metric* button (3). Quality of plans are displayed through a real-time chart in the *Charts* section (7). We have chosen a standard bar chart as the most appropriate representation for metric values. The *Plans Generation Tree* section (9) encodes a "plan hierarchy". The root of the hierarchy is the first plan generated by the planner. Each generated alternative plan saved by the user via the *Save Plan* button (6) is automatically inserted into the hierarchy as a child of the current plan. Each node in the hierarchy contains: the plan ID, its cost, the action suggested by the user and the action replaced. The tree allows an easy navigation between the plans the user is exploring. Finally the *Save Workspace* button (3) allows the user to save the current workspace.

6 Conclusions

We have presented a methodology for providing explanations for the decisions made by the planner. The core idea is to allow the user to explore alternative actions within plans, generate a set of new plans, and then compare their costs. An explanation for a plan found by the planner is provided by showing that the plan is no worse than the alternative plan the user would expect. The methodology is implemented in the new XAI-PLAN framework, which is integrated in ROSPlan, and provides a user-centered interface. Future work will explore a number of exciting issues such as temporal choices and context identification, and will feature a user study to assess the impact of explanations in trust and confidence.

Acknowledgements

We thank David Aha, Maria Fox, and Derek Long for their very useful comments and suggestions.

References

[Bidot *et al.*, 2010] Julien Bidot, Susanne Biundo, Tobias Heinroth, Wolfgang Minker, Florian Nothdurft, and Bernd Schattenberg. Verbal plan explanations for hybrid planning. In *MKWI*, 2010.

[Cashmore *et al.*, 2015] Michael Cashmore, Maria Fox, Derek Long, Daniele Magazzeni, Bram Ridder, Arnau Carrera, Narcis Palomeras, Natalia Hurtos, and Marc Carreras. ROSPlan: Planning in the robot operating system. In *ICAPS*, 2015.

[Cashmore *et al.*, 2018] Michael Cashmore, Maria Fox, Derek Long, Daniele Magazzeni, and Bram Ridder. Opportunistic planning in autonomous underwater missions. *IEEE Trans. Automation Science and Engineering*, 15(2):519–530, 2018.

[Chakraborti *et al.*, 2017] T. Chakraborti, S. Sreedharan, Y. Zhang, and S. Kambhampati. Plan explanations as model reconciliation: Moving beyond explanation as soliloquy. In *IJCAI*, 2017.

[Fox *et al.*, 2017] Maria Fox, Derek Long, and Daniele Magazzeni. Explainable planning. *IJCAI-17 Workshop on Explainable AI*, 2017.

[Hayes and Shah, 2017] Bradley Hayes and Julie A. Shah. Improving robot controller transparency through autonomous policy explanation. In *HRI*, 2017.

[Katz *et al.*, 2018] Michael Katz, Shirin Sohrabi, Octavian Udrea, and Dominik Winterer. A novel iterative approach to top-k planning. In *ICAPS*, 2018.

[Langley *et al.*, 2017] Pat Langley, Ben Meadows, Mohan Sridharan, and Dongkyu Choi. Explainable agency for intelligent autonomous systems. In *AAAI*, 2017.

[Lipovetzky *et al.*, 2014] Nir Lipovetzky, Christina N. Burt, Adrian R. Pearce, and Peter J. Stuckey. Planning for mining operations with time and resource constraints. In *ICAPS*, 2014.

[Morris *et al.*, 2015] Robert Morris, Mai Lee Chang, Ronald Archer, Ernest Vincent Cross II, Shelby Thompson, Jerry L. Franke, Robert Christopher Garrett, Waqar Malik, Kerry McGuire, and Garrett Hemann. Self-driving aircraft towing vehicles: A preliminary report. In *AAAI Workshop on AI for Transportation*, 2015.

[Nguyen *et al.*, 2012] Tuan Nguyen, Minh Do, Alfonso Gerevini, Ivan Serina, Biplav Srivastava, and Subbarao Kambhampati. Generating diverse plans to handle unknown and partially known user preferences. *Artificial Intelligence*, 2012.

[Piacentini *et al.*, 2016] Chiara Piacentini, Daniele Magazzeni, Derek Long, Maria Fox, and Chris Dent. Solving realistic unit commitment problems using temporal planning: Challenges and solutions. In *ICAPS*, 2016.

[Piotrowski *et al.*, 2016] Wiktor Mateusz Piotrowski, Maria Fox, Derek Long, Daniele Magazzeni, and Fabio Mercorio. Heuristic planning for PDDL+ domains. In *Proceedings of IJCAI*, 2016.

[Rosenthal *et al.*, 2016] Stephanie Rosenthal, Sai P. Selvaraj, and Manuela M. Veloso. Verbalization: Narration of autonomous robot experience. In *IJCAI*, 2016.

[Seegebarth *et al.*, 2012] Bastian Seegebarth, Felix Müller, Bernd Schattenberg, and Susanne Biundo. Making hybrid plans more clear to human users - A formal approach for generating sound explanations. In *ICAPS*, 2012.

[Smith, 2012] David Smith. Planning as an iterative process. In *AAAI*, 2012.

[Sohrabi *et al.*, 2011] Shirin Sohrabi, Jorge A. Baier, and Sheila A. McIlraith. Preferred explanations: Theory and generation via planning. In *AAAI*, 2011.

[Sreedharan *et al.*, 2018] Sarath Sreedharan, Tathagata Chakraborti, and Subbarao Kambhampati. Handling model uncertainty and multiplicity in explanations via model reconciliation. In *ICAPS*, 2018.

[Thiébaux *et al.*, 2013] Sylvie Thiébaux, Carleton Coffrin, Hassan L. Hijazi, and John K. Slaney. Planning with MIP for supply restoration in power distribution systems. In *IJCAI*, 2013.

[Vallati *et al.*, 2016] Mauro Vallati, Daniele Magazzeni, Bart De Schutter, Lukás Chrpa, and Thomas Leo McCluskey. Efficient macroscopic urban traffic models for reducing congestion: A PDDL+ planning approach. In *AAAI*, 2016.

[Zhang *et al.*, 2017] Y. Zhang, S. Sreedharan, A. Kulkarni, T. Chakraborti, H. Zhuo, and S. Kambhampati. Plan explicability and predictability for robot task planning. In *ICRA*, 2017.

Human-Aware Planning Revisited: *A Tale of Three Models*[*]

Tathagata Chakraborti and **Sarath Sreedharan** and **Subbarao Kambhampati**
Arizona State University, Tempe, AZ 85281 USA
{ tchakra2, ssreedh3, rao } @ asu.edu

Abstract

Human-aware planning requires an agent to be aware of the mental model of the humans, in addition to their physical or capability model. This not only allows an agent to envisage the desired roles of the human in a joint plan but also anticipate how its plan will be *perceived* by the latter. The human mental model becomes especially useful in the context of an explainable planning (XAIP) agent since an explanatory process cannot be a soliloquy, i.e. it must incorporate the human's beliefs and expectations of the planner. In this paper, we survey our recent efforts in this direction.

Cognitive AI teaming [Chakraborti *et al.*, 2017a] requires a planner to perform argumentation over a set of models during the plan generation process. This is illustrated in Figure 1. Here, $\mathcal{M}^R$ is the model of the agent embodying the planner (e.g. a robot), and $\mathcal{M}^H$ is the model of the human in the loop. Further, $\mathcal{M}^R_h$ is the model the human thinks the robot has, and $\mathcal{M}^H_r$ is the model that the robot thinks the human has. Finally, $\tilde{\mathcal{M}}^R_h$ is the robot's approximation of $\mathcal{M}^R_h$; for the rest of the paper we will be using $\mathcal{M}^R_h$ to refer to both since, for all intents and purposes, this is all the robot has access to. Note that the *human mental model* $\mathcal{M}^R_h$ is in addition to the (robot's belief of the) *human model* $\mathcal{M}^H_r$ traditionally encountered in human-robot teaming (HRT) settings and is, in essence, the fundamental thesis of the recent works on *plan explanations* [Chakraborti *et al.*, 2017b] and *explicable planning* [Zhang *et al.*, 2017]. The need for explicable planning or plan explanations occurs when the models – $\mathcal{M}^R$ and $\mathcal{M}^R_h$ – diverge so that the optimal plans in the respective models may not be the same and hence *optimal behavior of the robot in its own model is inexplicable to the human*. This is also true for discrepancies between $\mathcal{M}^H$ and $\mathcal{M}^H_r$ when the robot might reveal unrealistic expectations of the human in a joint plan.

An explainable planning (XAIP) agent [Fox *et al.*, 2017; Langley *et al.*, 2017] should be able to able to deal with such model differences and participate in explanatory dialog with the human such that both of them can be on the same page during a collaborative activity. This is referred to as *model reconciliation* [Chakraborti *et al.*, 2017b] and forms the core of the explanatory process of an XAIP agent. In this paper, we look at the scope of problems engendered by this multi-model setting and describe the recent work in this direction. Specifically –

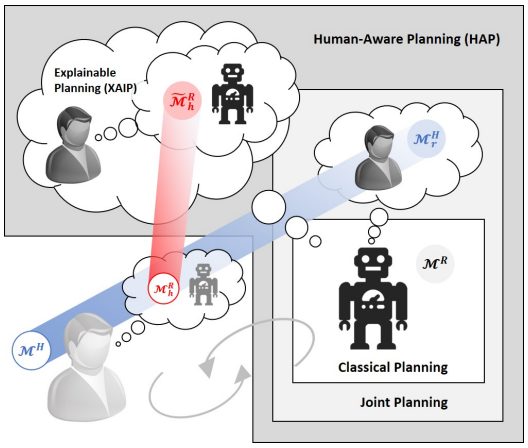


Figure 1: Argumentation over multiple models during the deliberative process of a human-aware planner (e.g. robot).

- We outline the scope of behaviors engendered by human-aware planning, including joint planning as studied in teaming using the *human model*, as well as explicable planning with the *human mental model*;
- We situate the plan explanation problem in the context of *perceived* inexplicability of the robot's plans or behaviors due to differences in these models;
  - We discuss how the plan explanation process can be seen as one of *model reconciliation* where $\mathcal{M}^R_h$ (and/or $\mathcal{M}^H_r$) is brought closer to $\mathcal{M}^R$ ($\mathcal{M}^H$);
  - We discuss how explicability and explanation costs can be traded off during plan generation;
  - We discuss how this process can be adapted to handle uncertainty or multiple humans in the loop;
  - We discuss results of a user study that testify to the usefulness of the model reconciliation process;
- We point to ongoing work in the space of abstractions and deception using the human mental model.

[*]An extended version of the paper can be found at http://rakaposhi.eas.asu.edu/ijcai_xai18_extended.pdf

Background

A Classical Planning Problem is a tuple $\mathcal{M} = \langle \mathcal{D}, \mathcal{I}, \mathcal{G} \rangle$ with domain $\mathcal{D} = \langle F, A \rangle$ – where F is a finite set of fluents that define a state $s \subseteq F$, and A is a finite set of actions – and initial and goal states $\mathcal{I}, \mathcal{G} \subseteq F$. Action $a \in A$ is a tuple $\langle c_a, \textit{pre}(a), \textit{eff}^{\pm}(a) \rangle$ where c_a is the cost, and $\textit{pre}(a), \textit{eff}^{\pm}(a) \subseteq F$ are the preconditions and add/delete effects, i.e. $\delta_{\mathcal{M}}(s, a) \models \perp$ *if* $s \not\models \textit{pre}(a)$; *else* $\delta_{\mathcal{M}}(s, a) \models s \cup \textit{eff}^{+}(a) \setminus \textit{eff}^{-}(a)$ where $\delta_{\mathcal{M}}(\cdot)$ is the transition function. The cumulative transition function is given by $\delta_{\mathcal{M}}(s, \langle a_1, a_2, \ldots, a_n \rangle) = \delta_{\mathcal{M}}(\delta_{\mathcal{M}}(s, a_1), \langle a_2, \ldots, a_n \rangle)$.

This forms the classical definition of a planning problem [Russell and Norvig, 2003] whose models are represented in the syntax of PDDL [McDermott *et al.*, 1998]. The solution to the planning problem is a sequence of actions or a (satisficing) *plan* $\pi = \langle a_1, a_2, \ldots, a_n \rangle$ such that $\delta_{\mathcal{M}}(\mathcal{I}, \pi) \models \mathcal{G}$. The cost of a plan π is given by $C(\pi, \mathcal{M}) = \sum_{a \in \pi} c_a$ if $\delta_{\mathcal{M}}(\mathcal{I}, \pi) \models \mathcal{G}$; ∞ otherwise. The cheapest plan $\pi^* = \arg\min_{\pi} C(\pi, \mathcal{M})$ is the (cost) optimal plan with cost $C^*_{\mathcal{M}}$.

In previous work [Nguyen *et al.*, 2017] we introduced an updated representation of planning problems in the form of *annotated* models to account for uncertainty or incompleteness over the definition of a classical planning model. In addition to the standard preconditions and effects associated with actions, it introduces the notion of *possible* preconditions and effects which may or may not be realized in practice.

An Incomplete (Annotated) Model is the tuple $\mathbb{M} = \langle \mathbb{D}, \mathbb{I}, \mathbb{G} \rangle$ with a domain $\mathbb{D} = \langle F, \mathbb{A} \rangle$ – where F is a finite set of fluents that define a state $s \subseteq F$, and $\mathbb{A}$ is a finite set of annotated actions – and annotated initial and goal states $\mathbb{I} = \langle \mathcal{I}^0, \mathcal{I}^+ \rangle$, $\mathbb{G} = \langle \mathcal{G}^0, \mathcal{G}^+ \rangle$; $\mathcal{I}^0, \mathcal{G}^0, \mathcal{I}^+, \mathcal{G}^+ \subseteq F$. Action $a \in \mathbb{A}$ is a tuple $\langle c_a, \textit{pre}(a), \widetilde{\textit{pre}}(a), \textit{eff}^{\pm}(a), \widetilde{\textit{eff}}^{\pm}(a) \rangle$ where c_a is the cost and, in addition to its *known* preconditions and add/delete effects $\textit{pre}(a), \textit{eff}^{\pm}(a), \subseteq F$ each action also contains *possible preconditions* $\widetilde{\textit{pre}}(a) \subseteq F$ containing propositions that action a *might* need as preconditions, and *possible add (delete) effects* $\widetilde{\textit{eff}}^{\pm}(a) \subseteq F$) containing propositions that the action a *might* add (delete, respectively) after execution. Similarly, $\mathcal{I}^0, \mathcal{G}^0$ (and $\mathcal{I}^+, \mathcal{G}^+$) are the known (and possible) parts of the initial and goal states.

Each possible condition $f \in \widetilde{\textit{pre}}(a) \cup \widetilde{\textit{eff}}^{\pm}(a)$ has a probability $p(f)$ associated with it denoting how likely it is to appear as a known condition in the ground truth model – i.e. $p(f)$ measures the confidence with which that condition has been learned. The sets of known and possible conditions in $\mathcal{M}$ are called $\mathbb{S}_k(\Gamma(\mathcal{M}))$ and $\mathbb{S}_p(\Gamma(\mathcal{M}))$ respectively. Here Γ is a mapping function that converts domain model conditions into propositions in a meta space [Chakraborti *et al.*, 2017b].

An *instantiation* of an annotated model $\mathbb{M}$ is a classical planning model where a subset of the possible conditions have been realized – given by the tuple $inst(\mathbb{M}) = \langle \mathcal{D}, \mathcal{I}, \mathcal{G} \rangle$ with domain $\mathcal{D} = \langle F, A \rangle$, initial and goal states $\mathcal{I} = \mathcal{I}^0 \cup \chi$; $\chi \subseteq \mathcal{I}^+$ and $\mathcal{G} = \mathcal{G}^0 \cup \chi$; $\chi \subseteq \mathcal{G}^+$ respectively, and action $A \ni a = \langle c_a, \textit{pre}(a) \leftarrow \textit{pre}(a) \cup \chi;\ \chi \subseteq \widetilde{\textit{pre}}(a), \textit{eff}^{\pm}(a) \leftarrow \textit{eff}^{\pm}(a) \cup \chi;\ \chi \subseteq \widetilde{\textit{eff}}^{\pm}(a) \rangle$. Given an annotated model with k possible conditions, there may be 2^k such instantiations, which forms its *completion set*.

The Likelihood $\mathcal{L}$ of an instantiation $inst(\mathbb{M})$ of the annotated model $\mathbb{M}$ is given by –

$$\mathcal{L}(inst(\mathbb{M})) = \prod_{f \in \mathbb{S}_p(\Gamma(\mathbb{M})) \wedge \mathbb{S}_k(\Gamma(inst(\mathbb{M})))} p(f) \times \prod_{f \in \mathbb{S}_p(\Gamma(\mathbb{M})) \setminus \mathbb{S}_k(\Gamma(inst(\mathbb{M})))} (1 - p(f))$$

Such models turn out to be especially useful for the representation and learning of human (mental) models from observations, where uncertainty after the learning process can be represented in terms of model annotations as in [Nguyen *et al.*, 2017; Bryce *et al.*, 2016]. Let $\mathbb{M}^R_H$ be the culmination of a model learning process and $\{\mathcal{M}^R_{h_i}\}_i$ be the completion set of $\mathbb{M}^R_H$. Note that one of these models – $g(\mathbb{M}^R_H)$ – is the actual ground truth (i.e. the human's real mental model).

Human-Aware Planning Revisited

The human-aware planning paradigm introduces the *mental model* of the human in the loop into a planner's deliberative process, in addition to the planner's own model in the classical sense and the robot's estimate of the *human model*.

A Human-Aware Planning (HAP) Problem is given by the tuple $\Psi = \langle \mathcal{M}^R, \mathcal{M}^H_r, \mathcal{M}^R_h \rangle$ where $\mathcal{M}^R = \langle D^R, \mathcal{I}^R, \mathcal{G}^R \rangle$ is the planner's model of a planning problem, while $\mathcal{M}^R_h = \langle D^R_h, \mathcal{I}^R_h, \mathcal{G}^R_h \rangle$ is the human's understanding of the same, and $\mathcal{M}^H_r = \langle D^H_r, \mathcal{I}^H_r, \mathcal{G}^H_r \rangle$ is the planner's belief of the human's capability model.

The solution to the human-aware planning problem is a joint plan [Chakraborti *et al.*, 2015] $\pi = \langle a_1, a_2, \ldots, a_n \rangle$; $a_i \in \{D^R \cup D^H_r\}$ such that $\delta_{\Psi}(\mathcal{I}^R \cup \mathcal{I}^H_r, \pi) \models \mathcal{G}^R \cup \mathcal{G}^H_r$. The robot's component in the plan is referred to as $\pi(R) = \langle a_i \mid a_i \in \pi \wedge D^R \rangle$; and similarly $\pi(H)$ for the human. Efforts to make planning more "human-aware" has largely focused on adapting $\pi(R)$ to meet the demands of $\pi(H)$ such as in [Alami *et al.*, 2006; 2014; Cirillo *et al.*, 2010; Koeckemann *et al.*, 2014; Tomic *et al.*, 2014; Cirillo, 2010; Chakraborti *et al.*, 2015; 2016] in the context of human-robot teams where a robot sacrifices optimality in its own model in favor of globally optimal joint plans. From the perspective of an XAIP agent, computation of the joint plan becomes more interesting when considering $\mathcal{M}^R_h$ as well, i.e. how $\pi(R)$ is *perceived* by the human. One solution is to be "explicable", i.e. make the robot conform to what the human expects of it.

Explicable Planning

An "explicable" solution to the human-aware planning problem is a plan π such that (1) it is executable (but may no longer be optimal) in the robot's model but is (2) "closer" to the expected plan in the human's model –

(1) $\delta_{\mathcal{M}^R}(\mathcal{I}^R, \pi) \models \mathcal{G}^R$; and

(2) $C(\pi, \mathcal{M}^R_h) \approx C^*_{\mathcal{M}^R_h}$.

Such a plan is referred to as explicable because the human can explain it in their current mental model. "Closeness" or distance to the expected plan is modeled here in terms of cost optimality, but in general this can be any metric such as plan similarity [Srivastava *et al.*, 2007; Nguyen *et al.*, 2012]. In existing literature [Zhang *et al.*, 2017; Kulkarni *et al.*, 2018a]

this has been achieved by modifying the search process so that the heuristic that guides the search is driven by the robot's knowledge of the human mental model. Such a heuristic can be either derived directly [Kulkarni *et al.*, 2018a] from the mental model or *learned* [Zhang *et al.*, 2017] through interactions in the form of affinity functions between plans and their purported goals. The solutions generated this way satisfy the planner's goal, as required by Condition (1), but are also biased towards the human's expectations as required by Condition (2) above.

It is interesting to note that, while mental modeling allows for human-awareness in the positive sense, it can also open up pathways for deception. Indeed, recent work [Kulkarni *et al.*, 2018b] has looked at how the concept of explicability can be flipped to obfuscate a robot's intentions from the observer.

Plan Explanations

The other approach would be to compute optimal plans in the planner's model (which may appear as inexplicable to the human) and provide an explanation of that plan in terms of the model differences – this is referred to as the process of *model reconciliation* [Chakraborti *et al.*, 2017b]. Although explanation of plans has been investigated in the past (c.f. [Kambhampati, 1990; Sohrabi *et al.*, 2011; Seegebarth *et al.*, 2012; Meadows *et al.*, 2013]), much of that work has involved the planner explaining its decisions with respect to its own model (i.e. current state, actions and goals) and assuming that this "*soliloquy*" also helps the human in the loop. While such a sanguine assumption may well be requited when the human is an expert "debugger" and is intimately familiar with the agent's innards, it is completely unrealistic in most human-AI interaction scenarios, where the humans may have a domain and task model that differs significantly from that used by the planner. We posit then that explanations should be seen as the robot's attempt to move the human's model to be in conformance with its own. The model reconciliation process thus forms the core of the explanation process for an XAIP agent and is thus the focus of the rest of the paper.

Our view of explanation as a model reconciliation process is supported by studies in the field of psychology which stipulate that explanations *"privilege a subset of beliefs, excluding possibilities inconsistent with those beliefs... can serve as a source of constraint in reasoning..."* [Lombrozo, 2006]. This is achieved in our case by the appropriate change in the expectation of the model that is believed to have engendered the plan in question. Further, authors in [Lombrozo, 2012] also underline that explanations are *"typically contrastive... the contrast provides a constraint on what should figure in a selected explanation..."* - this is especially relevant in order for an explanation to be self-contained and unambiguous. Hence the requirement of optimality, which not only ensures that the current plan is valid in the updated model, but is also better than other alternatives or foils [Miller, 2017].

The model reconciliation viewpoint can explain many phenomena in both explanation and transparency – e.g. the fact that well-performing, efficient teams require less, not more, explicit communication [Entin and Serfaty, 1999] and the characteristics of effective team debriefing [Tannenbaum and Cerasoli, 2013] after a mission or project.

The optimality criterion, and argumentation over the human mental model, makes the problem fundamentally different from model change algorithms in [Göbelbecker *et al.*, 2010; Herzig *et al.*, 2014; Eiter *et al.*, 2010; Bryce *et al.*, 2016; Porteous *et al.*, 2015] which focus more on the feasibility of plans or correctness of domains.

The Model Reconciliation Process

The explanation process, in response to a plan π that the robot has come up with and is perceived as inexplicable by the human, begins with the following question –

Q: Why not a different plan $\hat{\pi}$?

This questions can arise due to one or both of two causes –

- $\mathcal{M}_h^R$, i.e. the human's approximation of the robot model is wrong. Here, since it knows its own ground truth model, the robot can use an approximation of the human mental model (known unknown) to perform model reconciliation so that both of them are on the same page.
- $\mathcal{M}_r^H$, i.e. the robot's approximation of the human model is wrong. The above approach would not work here since the robot does not know what it does not know (i.e. the real human model is an unknown unknown). However, if the above approach fails to provide a satisfactory response from the human, then the robot can conclude it must be because of this and seek out more information to update its understanding of the human model.

For the first case, the model reconciliation approach would be to provide an (1) explanation or model update $\mathcal{E}$ such that the (2) robot optimal plan is (3) feasible and at least as good as the foil in the updated model, i.e.

(1) $\widehat{\mathcal{M}}_h^R \longleftarrow \mathcal{M}_h^R + \mathcal{E}$; and

(2) $C(\pi, \mathcal{M}^R) = C^*_{\mathcal{M}^R}$;

(3) $\delta_{\widehat{\mathcal{M}}_h^R}(\widehat{\mathcal{I}}_h^R, \pi) \models \widehat{\mathcal{G}}_h^R \ \wedge \ C(\pi, \widehat{\mathcal{M}}_h^R) < C(\hat{\pi}, \widehat{\mathcal{M}}_h^R)$.

The question can also be posed in the following form –

Q: Why plan π?

This, in essence, involves an implicit quantifier over all possible foils. Condition (3) above then must ensure that plan π is now also optimal in the updated mental model –

(3) $C(\pi, \widehat{\mathcal{M}}_h^R) = C^*_{\widehat{\mathcal{M}}_h^R}$.

In [Chakraborti *et al.*, 2017b] we explore different model reconciliation processes considering four characteristics –

R1. **Completeness -** Explanations of a plan should be able to be compared and contrasted against other alternatives, so that no better solution exists. We enforce this property by requiring that in the updated human model the plan being explained is optimal – i.e. Conditions (3).

R2. **Conciseness -** Explanation should be concise so that they are easily understandable to the explainee. Larger an explanation is, the harder it is for the human to incorporate that information into her deliberative process.

Table 1: Requirements for different types of explanations.

Explanation Type	R1	R2	R3	R4
Plan Patch Explanation	✗	✓	✗	✓
Model Patch Explanation	✓	✗	✓	✓
Minimally Complete Explanation	✓	✓	✗	?
Minimally Monotonic Explanation	✓	✓	✓	?

R3. **Monotonicity -** This ensures that remaining model differences cannot change the completeness of an explanation, i.e. all aspects of the model that engendered the plan have been reconciled. This thus subsumes completeness and requires more detailed explanations.

R4. **Computability -** While conciseness deals with how easy it is for the explainee to understand an explanation, computability measures the ease of computing the explanation from the point of view of the planner.

A Minimally Complete Explanation (MCE) is the shortest explanation that satisfies conditions (1) and (2).

A Minimally Monotonic Explanation (MME) is the shortest explanation that is both complete and monotonic.

A Plan Patch Explanation (PPE) only includes (all the) model updates pertaining to actions in the plan π.

A Model Patch Explanation (MPE) includes all the model updates $|\mathcal{M}^R \Delta \mathcal{M}_h^R|$.

The requirements outlined above are thus often at odds - an explanation that is very easy to compute may be very hard to comprehend (c.f. Table 1). A detailed account of these explanations can be found in [Chakraborti *et al.*, 2017b]; we will concentrate on MCEs for the rest of the paper.

Remark Note that during model reconciliation process, the robot model need not be the ground truth. However, the robot can only explain with respect to what it believes to be true. This can, of course, be wrong and be refined iteratively through interaction with the human, as demonstrated in a decision support setting in [Sengupta *et al.*, 2017b].

Remark We insisted that explanations must be compatible with the planners model. If this is relaxed, it allows the planner to generate "explanations" that it knows are not true, and thus deceive the human. In recent work [Chakraborti and Kambhampati, 2018], we have shown that participants in a study were generally positive towards lying for the greater good especially when those actions would not be determined by their teammate, but is loath to suspend normative behavior, robot or not, in the event that they would be caught in that act unless the robot is the recipient of the misinformation.

Remark While in this line of work, we are concerned more with the generation of the *content* of explanations rather than the actual delivery of this information, there has been some recent work to this end. Depending on the type of interaction between the planner and the human, this can be achieved by means of natural language dialog [Perera *et al.*, 2016], in the form of a graphical user interface [Sengupta *et al.*, 2017b; Chakraborti *et al.*, 2018b] or even in mixed-reality interfaces [Chakraborti *et al.*, 2018e; 2018c].

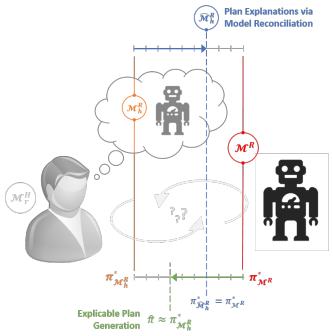


Figure 2: Balancing explicability and explanations in HAP.

How to chose between Explicability/Explanations?

Explanations and explicability are intrinsically related in an agent's deliberative process (c.f. Figure 2) – it can generate a explicable plan to the best of its ability or it can provide explanations whenever required, or it can even opt for a combination of both if the expected human plan is too costly in its own model (e.g. the human might not be aware of some safety constraints) or the cost of communication overhead for explanations is too high (e.g. limited communication bandwidth). In [Chakraborti *et al.*, 2018d], we looked at the problem of attaining the sweet spot between plan explanations and explicability during the decision making process.

From the perspective of design of autonomy, the explicability versus explanations trade-off has two interesting implications – (1) the agent can now not only explain but also *plan* in the multi-model setting with the trade-off between compromise on its optimality and possible explanations in mind; and (2) the argumentation process is known to be a crucial function of the reasoning capabilities of humans [Mercier and Sperber, 2010], and now by extension of autonomous agents as well, as a result of these algorithms that incorporate the explanation generation process into the decision making process itself. General argumentation frameworks for resolving disputes over plans have indeed been explored before [Belesiotis *et al.*, 2010; Emele *et al.*, 2011]. Other forms of argumentation [Russell and Wefald, 1991] has been aimed at meta-level reasoning of resource usage or cost of solutions. Our work can be seen as the specific case where the argumentation process is over a set of constraints that trade-off the quality of a plan and the cost of explaining it. This is the first of its kind algorithm that can achieve this.

The result of a trade off in the relative cost of explicability and explanations during the plan generation process is a plan π and an explanation $\mathcal{E}$ such that (1) π is executable in the robot's model, and with the explanation (2) in the form of model updates it is (3) optimal in the updated human model while (4) the cost (length) of the explanations, and the cost of deviation from optimality in its own model to be explicable to the human, is traded off according to a constant α –

(1) $\delta_{\mathcal{M}^R}(\mathcal{I}^R, \pi) \models \mathcal{G}^R$;

(2) $\widehat{\mathcal{M}}_h^R \longleftarrow \mathcal{M}_h^R + \mathcal{E}$;

(3) $C(\pi, \widehat{\mathcal{M}}_h^R) = C^*_{\widehat{\mathcal{M}}_h^R}$; and

(4) $\pi = \arg\min_\pi \{ |\mathcal{E}| + \alpha \times | C(\pi, \mathcal{M}^R) - C^*_{\mathcal{M}^R} | \}$.

Higher values of α will produce plans that require more explanation while lower α will generate more explicable plans. Thus, with the help of this hyperparameter, the agent can deliberate over the trade-off in the costs it incurs in being explicable to the human (second minimizing term) versus explaining its decisions (first minimizing term). Note that this trade-off is irrespective of the cognitive burden of those decisions on the human. For example, a robot in a collapsed building during a search and rescue task, may have limited bandwidth for communication and hence prefer to be explicable.

What happens if the mental model is unknown?

The model reconciliation process described above is only feasible if inconsistencies of the robot model with the human mental model are known precisely. Although we made this assumption so far as a first step towards formalizing the model reconciliation process, this can be hard to achieve in practice. Instead, the agent may end up having to explain its decisions with respect to a *set of possible models* which is its best estimation of the human's knowledge state learned in the process of interactions. In this situation, the robot can try to compute MCEs for each possible configuration. However, this can result in situations where the explanations computed for individual models independently are not consistent across all possible target domains. Thus, in the case of model uncertainty, such an approach cannot guarantee that the resulting explanation will be acceptable. Instead, we want to find an explanation such that $\forall i\ \pi^*_{\widehat{\mathcal{M}}^R_{h_i}} \equiv \pi^*_{\mathcal{M}^R}$.

This is a single model update that makes the given plan optimal (and hence explained) in all the updated domains (or in all possible domains). At first glance, it appears that such an approach, even though desirable, might turn out to be prohibitively expensive especially since solving for a *single* MCE involves search in the model space where each search node is an optimal planning problem [Chakraborti *et al.*, 2017b]. However, it turns out that the same search strategy can be employed here as well by representing the human mental model as an *annotated* model [Sreedharan *et al.*, 2018a]. Condition (3) for an MCE now becomes –

(3) $C(\pi, g(\mathbb{M}^R_h)) = C^*_{g(\mathbb{M}^R_h)}$

This is hard to achieve since it is not known which is the actual mental model of the human. So we want to preserve the optimality criterion for all (or as many) instantiations of the incomplete estimation of the explainee's mental model. Keeping this in mind, we define *robustness* of an explanation for an incomplete mental models as the probability mass of models where it is a valid explanation.

Robustness of an explanation $\mathcal{E}$ is given by –

$$R(\mathcal{E}) = \sum_{inst(\widehat{\mathcal{M}}^R_h) \text{ s.t. } C(\pi, inst(\widehat{\mathcal{M}}^R_h)) = C^*_{inst(\widehat{\mathcal{M}}^R_h)}} \mathcal{L}(inst(\widehat{\mathcal{M}}^R_h))$$

A Conformant Explanation is such that $R(\mathcal{E}) = 1$.

This ensures that the given plan is explained in all the models in the completion set of the human mental model.

Note that conformant explanations can contain superfluous information – i.e. asking the human to remove non-existent conditions or add existing ones. In the previous example, the second explanation (regarding the need of the hand to clear rubble) was already known to the human and was thus superfluous information. Such redundant information can be annoying and may end up reducing the human's trust in the robot. This can be avoided by –

- Increasing the cost of model updates involving uncertain conditions relative to those involving known preconditions or effects. This ensures that the search prefers explanations that contain known conditions. By definition, such explanations will not have superfluous information.
- However, such explanations may not exist. Instead, we can convert conformant explanations into *conditional* ones by turning each model update for an annotated condition into a question and only provide an explanation if the human's response warrants it.

Thus, one way of removing superfluous explanations is to engage the human in conversation and reduce the size of the completion set (i.e. uncertainty over the mental model) .

A Conditional Explanation is represented by a policy that maps the annotated model (represented by a $\mathcal{M}_{min}$ and $\mathcal{M}_{max}$ model pair) to either a question regarding the existence of a condition in the human ground model or a model update request. The resultant annotated model is produced, by either applying the model update directly into the current model or by updating the model to conform to human's answer regarding the existence of the condition.

Note that in asking questions such as these, the robot is trying to exploit the human's (lack of) knowledge of the problem in order to provide more concise explanations. This can be construed as a case of lying by omission and can raise interesting ethical considerations [Chakraborti and Kambhampati, 2018]. Humans, during an explanation process, tend to undergo this same "selection" process [Miller, 2017] as well in determining which of the many reasons that could explain an event is worth highlighting. It is worthwhile investigating similar behavior for autonomous agents.

Anytime Explanations Since dealing with model uncertainty can be computationally expensive, we relax the minimality requirement and introduce an anytime depth first explanation generation algorithm. This is explained in detail in [Sreedharan *et al.*, 2018a].

What if there are multiple humans in the loop?

While generating explanations for a *set of models*, the robot is essentially trying to cater to multiple human models at the same time. We posit then that the same approaches can be adopted to situations when there are *multiple humans* in the loop instead of a single human whose model is not known with certainty. As before, computing separate explanations [Chakraborti *et al.*, 2017b] for each agent can result in situations where the explanations computed for individual models independently are not consistent across all the possible target domains. In the case of multiple teammates being explained to, this may cause confusion and loss of trust, especially in teaming with humans who are known [Cooke *et al.*, 2013] to rely on shared mental models. Thus *conformant explanations*

can find useful applications in dealing with not only model uncertainty but also model multiplicity.

In order to do this, from the set of target human mental models we construct an annotated model so that *the preconditions and effects that appear in all target models become necessary ones, and those that appear in just a subset are possible ones*. As before, we find a single explanation that is a satisfactory explanation for the entire set of models, without having to repeat the standard MRP process over all possible models while coming up with an explanation that can satisfy all of them and thus establish common ground [Chakraborti *et al.*, 2018c; Sreedharan *et al.*, 2018a].

How to model human expertise?

Most of the above discussion has focused on generating explanations in cases where both the human and the robot understands the task at the same granularity. Applying model reconciliation without acknowledging the difference in the level of "expertise" can lead to confusion and information overload. Indeed, explanation generation techniques for machine learning systems have explicitly modeled this difference [Ribeiro *et al.*, 2016; 2018].

In [Sreedharan *et al.*, 2018b], authors have looked at ways of generating explanations when the human has access to only an abstract version of the model of the robot. Specifically, they focus on state abstractions where the abstract model was formed by projecting out a certain subset of state fluents [Srivastava *et al.*, 2016], though the principles are likely to carry over to other types of abstraction as well (e.g. temporal abstractions of the types discussed in [Marthi *et al.*, 2007]). Since the abstract model may be logically weaker, the human may incorrectly believe that an optimal plan suggested by the planner is suboptimal. When presented with the plan π, the human can respond by either presenting a set of *foils* F. In such cases, the *explanation* takes the form of information about a set of state properties which when introduced into the human model resolves or invalidates the set of foils. Thus, the explanation can be uniquely represented by a sequence of propositions $\epsilon = \langle p_1, .., p_k \rangle$ as follows –

(1) A set of foils $F = \{\pi_1, .., \pi_m\}$ such that $\forall \pi \in F,\ \delta_{\mathcal{M}^R}(\mathcal{I}^R, \pi) \not\models \mathcal{G}^R$ and $\delta_{\mathcal{M}_h^R}(\mathcal{I}_h^R, \pi) \not\models \mathcal{G}_h^R$

(2) An explanation $\epsilon = \langle p_1, ..., p_k \rangle$, such that $\widehat{\mathcal{M}}_h^R \longleftarrow \mathcal{M}_h^R + \mathcal{E}$;

(3) $\forall \pi \in F,\ \delta_{\widehat{\mathcal{M}}_h^R}(\mathcal{I}_h^R, \pi) \not\models \mathcal{G}_h^R$

One of the main challenges with this method is the uncertainty about the human model. To address this, the authors build a lattice of possible models from the task model called *model lattice* ($\mathcal{L}$) as shown in Figure 3. The lattice consists of possible abstractions of the concrete task model and an edge exists between two models if there exists a single predicate that can be projected from one model to create the other. The foils are used to estimate the possible human mental model.

How do humans reconcile models?

The design of "human-aware" algorithms is, of course, incomplete without evaluations of the same with actual humans in the loop. Thus, in the final part of this discussion, we will report on the the salient findings from a controlled user study

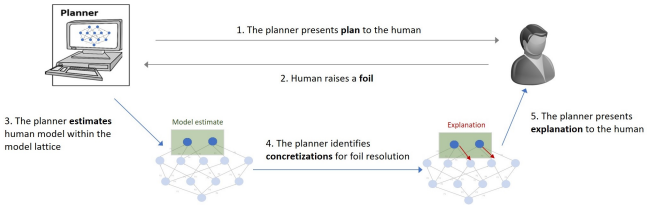


Figure 3: Hierarchical explanation generation approach.

we undertook recently in order to evaluate the usefulness of the model reconciliation approach. A detailed report of the study can be read at [Chakraborti *et al.*, 2018a].

Experimental Setup During the study, participants had to identify if a given plan (which may be optimal in the robot model or explicable or even balanced) looks optimal or satisficing to them. If unsure, they can ask for an explanation. The explanations provided were of the types described before.

Study-1 In the first set of experiments, participants assumed the role of the explainer. It was found that, when left to themselves, they generated explanations of the type MPE or (if communication was restricted) MCE. Further, in subjective responses, they considered model reconciliation as necessary and sufficient for the explanation process.

Study-2 Here, participants assumed the role of the explainer, and had to identify, on the basis of explanations provided the quality of the given plan. We found that the participants were indeed able to distinguish between optimal plans (when provided with MCEs or MPEs) and (perceived) satisficing plans (when provided with PPEs) and were in general overwhelmingly in favor of model reconciliation as an effective form of explanation. We further found that explicable plans indeed reduced the call of explanations, while balanced plans preserved their outlook towards the explanations while allowing the robot to trade-off its communication cost with the optimality of its plans.

Work in Progress

So far we have have not considered differences in the cost function which, though falls under the scope of model differences to be explained, can introduce interesting challenges to the model reconciliation problem. It may be possible to learn such functions through interactions with the human as in [Zhang *et al.*, 2017; Kulkarni *et al.*, 2018a]. Further, we have not modeled the computational capability of the human which also affect the process – this can be potentially handled by modeling $\epsilon-$optimal humans or by considering top-K plans [Katz *et al.*, 2018] while checking the optimality condition during model space search. Finally, we do not consider system level constraints (such as time and resources) in the explanations which remain out of scope of this discussion.

In terms of applications, we are also exploring different settings for the adoption of the human-aware planning techniques introduced in the paper. This includes human-robot teaming [Chakraborti *et al.*, 2018d] and also recently in decision support [Sengupta *et al.*, 2017a; Chakraborti *et al.*, 2018c] and intelligent tutoring systems [Grover *et al.*, 2018].

Acknowledgements This research is supported in part by the AFOSR grant FA9550-18-1-0067, the ONR grants N00014-16-1-2892, N00014-13-1-0176, N00014-13-1-0519, N00014-15-1-2027, N00014-18-1-2442 and the NASA grant NNX17AD06G. The first author is also supported by the IBM Ph.D. Fellowship from 2016-18.

References

[Alami *et al.*, 2006] Rachid Alami, Aurélie Clodic, Vincent Montreuil, Emrah Akin Sisbot, and Raja Chatila. Toward Human-Aware Robot Task Planning. In *AAAI Spring Symposium*, 2006.

[Alami *et al.*, 2014] Rachid Alami, Mamoun Gharbi, Benjamin Vadant, Raphaël Lallement, and Adolfo Suarez. On human-aware task and motion planning abilities for a teammate robot. In *Human-Robot Collaboration for Industrial Manufacturing Workshop, RSS*, 2014.

[Belesiotis *et al.*, 2010] Alexandros Belesiotis, Michael Rovatsos, and Iyad Rahwan. Agreeing on plans through iterated disputes. In *AAMAS*, pages 765–772, 2010.

[Bryce *et al.*, 2016] Dan Bryce, J. Benton, and Michael W. Boldt. Maintaining Evolving Domain Models. In *IJCAI*, 2016.

[Chakraborti and Kambhampati, 2018] T. Chakraborti and S. Kambhampati. Algorithms for the Greater Good! On Mental Modeling and Acceptable Symbiosis in Human-AI Collaboration. *ArXiv e-prints*, January 2018.

[Chakraborti *et al.*, 2015] T. Chakraborti, G. Briggs, K. Talamadupula, Yu Zhang, M. Scheutz, D. Smith, and S. Kambhampati. Planning for serendipity. In *IROS*, 2015.

[Chakraborti *et al.*, 2016] Tathagata Chakraborti, Yu Zhang, David Smith, and Subbarao Kambhampati. Planning with Resource Conflicts in Human-Robot Cohabitation. In *AAMAS*, 2016.

[Chakraborti *et al.*, 2017a] Tathagata Chakraborti, Subbarao Kambhampati, Matthias Scheutz, and Yu Zhang. AI Challenges in Human-Robot Cognitive Teaming. *arXiv preprint arXiv:1707.04775*, 2017.

[Chakraborti *et al.*, 2017b] Tathagata Chakraborti, Sarath Sreedharan, Yu Zhang, and Subbarao Kambhampati. Plan Explanations as Model Reconciliation: Moving Beyond Explanation as Soliloquy. In *IJCAI*, 2017.

[Chakraborti *et al.*, 2018a] T. Chakraborti, S. Sreedharan, S. Grover, and S. Kambhampati. Plan Explanations as Model Reconciliation – An Empirical Study. *ArXiv e-prints*, February 2018.

[Chakraborti *et al.*, 2018b] Tathagata Chakraborti, Fadnis P. Kshitij, Kartik Talamadupula, Mishal Dholakia, Biplav Srivastava, Jeffrey O Kephart, and Rachel KE Bellamy. Visualizations for an explainable planning agent. *ICAPS UISP*, 2018.

[Chakraborti *et al.*, 2018c] Tathagata Chakraborti, Sailik Sengupta, and Subbarao Kambhampati. MA-RADAR – A Mixed-Reality Interface for Collaborative Decision Making. *ICAPS UISP*, 2018.

[Chakraborti *et al.*, 2018d] Tathagata Chakraborti, Sarath Sreedharan, and Subbarao Kambhampati. Balancing Explicability and Explanation in Human-Aware Planning. In *AAMAS*, 2018.

[Chakraborti *et al.*, 2018e] Tathagata Chakraborti, Sarath Sreedharan, Anagha Kulkarni, and Subbarao Kambhampati. Projection-Aware Task Planning and Execution for Human-in-the-Loop Operation of Robots. In *IROS*, 2018.

[Cirillo *et al.*, 2010] Marcello Cirillo, Lars Karlsson, and Alessandro Saffiotti. Human-aware Task Planning: An Application to Mobile Robots. *ACM Transactions on Intelligent Systems and Technology*, 2010.

[Cirillo, 2010] Marcello Cirillo. *Planning in inhabited environments: human-aware task planning and activity recognition*. PhD thesis, Örebro university, 2010.

[Cooke *et al.*, 2013] Nancy J Cooke, Jamie C Gorman, Christopher W Myers, and Jasmine L Duran. Interactive team cognition. *Cognitive science*, 37(2):255–285, 2013.

[Eiter *et al.*, 2010] Thomas Eiter, Esra Erdem, Michael Fink, and Ján Senko. Updating action domain descriptions. *Artificial intelligence*, 2010.

[Emele *et al.*, 2011] Chukwuemeka D Emele, Timothy J Norman, and Simon Parsons. Argumentation strategies for plan resourcing. In *AAMAS*, 2011.

[Entin and Serfaty, 1999] Elliot E Entin and Daniel Serfaty. Adaptive team coordination. *Human factors*, 41(2):312–325, 1999.

[Fox *et al.*, 2017] Maria Fox, Derek Long, and Daniele Magazzeni. Explainable Planning. In *IJCAI XAI*, 2017.

[Göbelbecker *et al.*, 2010] Moritz Göbelbecker, Thomas Keller, Patrick Eyerich, Michael Brenner, and Bernhard Nebel. Coming up with good excuses: What to do when no plan can be found. 2010.

[Grover *et al.*, 2018] Sachin Grover, Tathagata Chakraborti, and Subbarao Kambhampati. What can Automated Planning do for Intelligent Tutoring Systems? In *ICAPS SPARK*, 2018.

[Herzig *et al.*, 2014] Andreas Herzig, Viviane Menezes, Leliane Nunes de Barros, and Renata Wassermann. On the revision of planning tasks. In *ECAI*, 2014.

[Kambhampati, 1990] Subbarao Kambhampati. A classification of plan modification strategies based on coverage and information requirements. In *AAAI 1990 Spring Symposium on Case Based Reasoning*, 1990.

[Katz *et al.*, 2018] Michael Katz, Shirin Sohrabi, Octavian Udrea, and Dominik Winterer. A Novel Iterative Approach to Top-k Planning. In *ICAPS*, 2018.

[Koeckemann *et al.*, 2014] Uwe Koeckemann, Federico Pecora, and Lars Karlsson. Grandpa Hates Robots - Interaction Constraints for Planning in Inhabited Environments. In *AAAI*, 2014.

[Kulkarni *et al.*, 2018a] Anagha Kulkarni, Tathagata Chakraborti, Yantian Zha, Satya Gautam Vadlamudi, Yu Zhang, and Subbarao Kambhampati. Explicable

Robot Planning as Minimizing Distance from Expected Behavior. In *ICAPS XAIP*, 2018.

[Kulkarni *et al.*, 2018b] Anagha Kulkarni, Siddharth Srivastava, and Subbarao Kambhampati. Implicit robot-human communication in adversarial and collaborative environments. In *ICAPS PlanRob*, 2018.

[Langley *et al.*, 2017] Pat Langley, Ben Meadows, Mohan Sridharan, and Dongkyu Choi. Explainable Agency for Intelligent Autonomous Systems. In *AAAI/IAAI*, 2017.

[Lombrozo, 2006] Tania Lombrozo. The Structure and Function of Explanations . *Trends in Cognitive Sciences*, 10(10):464 – 470, 2006.

[Lombrozo, 2012] Tania Lombrozo. Explanation and abductive inference. *Oxford handbook of thinking and reasoning*, pages 260–276, 2012.

[Marthi *et al.*, 2007] Bhaskara Marthi, Stuart J Russell, and Jason Andrew Wolfe. Angelic semantics for high-level actions. In *ICAPS*, pages 232–239, 2007.

[McDermott *et al.*, 1998] Drew McDermott, Malik Ghallab, Adele Howe, Craig Knoblock, Ashwin Ram, Manuela Veloso, Daniel Weld, and David Wilkins. PDDL-the planning domain definition language. 1998.

[Meadows *et al.*, 2013] Ben Leon Meadows, Pat Langley, and Miranda Jane Emery. Seeing beyond shadows: Incremental abductive reasoning for plan understanding. In *AAAI Workshop: Plan, Activity, and Intent Recognition*, volume 13, page 13, 2013.

[Mercier and Sperber, 2010] Hugo Mercier and Dan Sperber. Why Do Humans Reason? Arguments for an Argumentative Theory. *Behavioral and Brain Sciences*, 2010.

[Miller, 2017] Tim Miller. Explanation in Artificial Intelligence: Insights from the Social Sciences. *CoRR*, abs/1706.07269, 2017.

[Nguyen *et al.*, 2012] Tuan Anh Nguyen, Minh Do, Alfonso Emilio Gerevini, Ivan Serina, Biplav Srivastava, and Subbarao Kambhampati. Generating diverse plans to handle unknown and partially known user preferences. *Artificial Intelligence*, 2012.

[Nguyen *et al.*, 2017] Tuan Nguyen, Sreedharan, and Subbarao Kambhampati. Robust planning with incomplete domain models. *Artificial Intelligence*, 245:134 – 161, 2017.

[Perera *et al.*, 2016] Vittorio Perera, Sai P. Selvaraj, Stephanie Rosenthal, and Manuela Veloso. Dynamic Generation and Refinement of Robot Verbalization. In *RO-MAN*, 2016.

[Porteous *et al.*, 2015] Julie Porteous, Alan Lindsay, Jonathon Read, Mark Truran, and Marc Cavazza. Automated extension of narrative planning domains with antonymic operators. In *AAMAS*, pages 1547–1555, 2015.

[Ribeiro *et al.*, 2016] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Why should i trust you?: Explaining the predictions of any classifier. In *ACM SIGKDD*, pages 1135–1144. ACM, 2016.

[Ribeiro *et al.*, 2018] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Anchors: High-precision model-agnostic explanations. In *AAAI*, 2018.

[Russell and Norvig, 2003] Stuart Russell and Peter Norvig. *Artificial intelligence: a modern approach*. 2003.

[Russell and Wefald, 1991] Stuart Russell and Eric Wefald. Principles of metareasoning. *Artificial intelligence*, 49(1-3):361–395, 1991.

[Seegebarth *et al.*, 2012] Bastian Seegebarth, Felix Müller, Bernd Schattenberg, and Susanne Biundo. Making hybrid plans more clear to human users-a formal approach for generating sound explanations. In *ICAPS*, 2012.

[Sengupta *et al.*, 2017a] Sailik Sengupta, Tathagata Chakraborti, Sarath Sreedharan, and Subbarao Kambhampati. RADAR - A Proactive Decision Support System for Human-in-the-Loop Planning. In *ICAPS Workshop on User Interfaces for Scheduling and Planning*, 2017.

[Sengupta *et al.*, 2017b] Sailik Sengupta, Tathagata Chakraborti, Sarath Sreedharan, and Subbarao Kambhampati. Radar-a proactive decision support system for human-in-the-loop planning. In *AAAI Fall Symposium on Human-Agent Groups*, 2017.

[Sohrabi *et al.*, 2011] Shirin Sohrabi, Jorge A. Baier, and Sheila A. McIlraith. Preferred explanations: Theory and generation via planning. In *AAAI-11*, pages 261–267, San Francisco, USA, August 2011.

[Sreedharan *et al.*, 2018a] Sarath Sreedharan, Tathagata Chakraborti, and Subbarao Kambhampati. Handling Model Uncertainty and Multiplicity in Explanations as Model Reconciliation. In *ICAPS*, 2018.

[Sreedharan *et al.*, 2018b] Sarath Sreedharan, Siddharth Srivastava, and Subbarao Kambhampati. Hierarchical expertise-level modeling for user specific robot-behavior explanations. *arXiv preprint arXiv:1802.06895*, 2018.

[Srivastava *et al.*, 2007] Biplav Srivastava, Tuan Anh Nguyen, Alfonso Gerevini, Subbarao Kambhampati, Minh Binh Do, and Ivan Serina. Domain independent approaches for finding diverse plans. In *IJCAI*, 2007.

[Srivastava *et al.*, 2016] Siddharth Srivastava, Stuart J Russell, and Alessandro Pinto. Metaphysics of planning domain descriptions. In *AAAI*, 2016.

[Tannenbaum and Cerasoli, 2013] Scott I Tannenbaum and Christopher P Cerasoli. Do team and individual debriefs enhance performance? a meta-analysis. *Human factors*, 55(1):231–245, 2013.

[Tomic *et al.*, 2014] Stevan Tomic, Federico Pecora, and Alessandro Saffiotti. Too Cool for School??? Adding Social Constraints in Human Aware Planning. In *Workshop on Cognitive Robotics (CogRob)*, 2014.

[Zhang *et al.*, 2017] Yu Zhang, Sreedharan, Anagha Kulkarni, Tathagata Chakraborti, Hankz Hankui Zhuo, and Subbarao Kambhampati. Plan Explicability and Predictability for Robot Task Planning. In *ICRA*, 2017.

Explanatory predictions with artificial neural networks and argumentation

Oana Cocarascu, Kristijonas Čyras, Francesca Toni
Imperial College London, UK, {oc511, k.cyras, f.toni}@imperial.ac.uk

Abstract

Data-centric AI has proven successful in several domains, but its outputs are often hard to explain. We present an architecture combining Artificial Neural Networks (ANNs) for feature selection and an instance of Abstract Argumentation (AA) for reasoning to provide effective predictions, explainable both dialectically and logically. In particular, we train an autoencoder to rank features in input examples, and select highest-ranked features to generate an AA framework that can be used for making and explaining predictions as well as mapped onto logical rules, which can equivalently be used for making predictions and for explaining. We show empirically that our method significantly outperforms ANNs and a decision-tree-based method from which logical rules can also be extracted.

1 INTRODUCTION

Data-centric AI has recently received a great deal of attention and shown success in several domains (e.g. see [LeCun *et al.*, 2015] for Artificial Neural Networks (ANNs)), but its predictions are notoriously hard to explain (e.g. see [Andrews *et al.*, 1995], again for ANNs). Argumentation has been increasingly studied in AI in the last two decades, mostly to deal with incomplete and conflicting information (see [Atkinson *et al.*, 2017] for a recent survey) but also to provide dialectical explanations for reasoning outputs of various kinds (e.g. see [Amgoud and Serrurier, 2007; Fan and Toni, 2015]). These explanations are dialectical in that they can be seen as debates between two opposing parties. We combine ANNs (of the 'autoencoder' variety [Hinton and Salakhutdinov, 2006]) and argumentation (of the 'Abstract Argumentation' (AA) variety [Dung, 1995]) to provide effective predictions that can be explained dialectically as well as logically, from automatically extracted rules. We call our methodology ANNA (for ANNs with argumentation).

Within ANNA, during training an autoencoder is used to rank features in input examples, as more or less representative. These training examples are labelled as belonging to one of two given classes (*outcomes*). The ranking is then used to select a subset of (highest-ranked) features so that the restriction of the training examples to these features is coherent (namely, with no two restrictions having the same features but different outcomes). The resulting restriction of the training examples is then mapped onto an AA framework (AAF), using the approach of [Čyras *et al.*, 2016a], and in turn this AAF is mapped onto a set of logical rules (of the 'logic programming' variety). In the AAF, the set of arguments consists of the (restrictions of) examples as well as a *default* argument (giving the default outcome in the absence of any information), and an argument attacks another if they have different outcomes and the former is a 'concise' superset of the latter. In the logic program, the default outcome is provable by default, and rules admit exceptions, obtained from examples.

The AAF and the logic program can be equivalently used (during testing) for prediction of the outcome for unlabelled examples. For a given example M, with restriction N to the set of features selected during training, the prediction is made as follows: (i) by adding to the AAF a new argument for N, attacking every argument in the AAF with features irrelevant to N, and determining whether the default argument is dialectically acceptable in the extended AAF, using the grounded semantics [Dung, 1995]; or, equivalently (ii) by adding to the logic program a fact (rule with an empty body) for every feature in N and determining whether the default outcome being provable is a logical consequence of the extended logic program, using semantics for logic programming [Apt and Bol, 1994]. The extended AAF and logic program can be used to explain the prediction, dialctically or logically, respectively.

We show empirically, using the dataset of [Dheeru and Karra Taniskidou, 2017], that ANNA significantly outperforms ANNs, while being less sensitive to the size of the training dataset, as well as a decision-tree-based method from which logical rules can also be extracted.

2 PRELIMINARIES

In this section we give essential background on Artificial Neural Networks (ANNs), argumentation and logic programming, of the kinds used within our method.

ANNs have been widely applied both in classification tasks and in dimensionality reduction, e.g. as in [Verikas and Bacauskiene, 2002]. Typically, dimensionality reduction refers to selecting a subset of the original feature set (selection) or mapping the initial data in M-dimensional space onto a K-dimensional space with $K < M$ (extraction). ANN-based

feature selection methods use multilayer perceptrons to determine which features are redundant [Gasca *et al.*, 2006] as well as autoencoders [Hinton and Salakhutdinov, 2006; Wang *et al.*, 2017; Han *et al.*, 2017]. These are unsupervised learning models based on ANNs which take a set of features as input and aim, through training, to reconstruct the inputs [Hinton and Salakhutdinov, 2006; Erhan *et al.*, 2010].

In this paper we use autoencoders for feature selection and generic ANNs for experimental comparison.

AA-CBR [Čyras *et al.*, 2016a; Čyras *et al.*, 2016b] is an Abstract Argumentation (AA)-driven and Case-Based Reasoning (CBR)-inspired model for reasoning with (past) cases (where a case is a set of features together with an outcome) to predict the outcomes of new cases.

Consider a fixed but otherwise arbitrary (possibly infinite) set $\mathbb{F}$ of *features*, and a set $\mathbb{O} = \{\delta, \overline{\delta}\}$ of two (distinct) *outcomes*, with δ called the *default outcome*. Then:

- a *case* is a pair (X, o) with $X \subseteq \mathbb{F}$ and $o \in \mathbb{O}$;
- a *case base* is a finite set $CB \subseteq \wp(\mathbb{F}) \times \mathbb{O}$ that is *coherent*, i.e. for $(X, o_X), (Y, o_Y) \in CB$, if $X = Y$ then $o_X = o_Y$;
- a *new case* is a pair $(N, ?)$ with $N \subseteq \mathbb{F}$ and ? indicating that the outcome is yet unknown.

To illustrate, we use the following example throughout.

Example 2.1. Let $\mathbb{F} = \{\mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{d}\}$, $CB = \{(\{\mathsf{a}\}, \overline{\delta}), (\{\mathsf{b}\}, \overline{\delta}), (\{\mathsf{a}, \mathsf{c}\}, \delta), (\{\mathsf{b}, \mathsf{d}\}, \overline{\delta})\}$ and $(N, ?) = (\{\mathsf{a}, \mathsf{d}\}, ?)$. It is easy to see that CB is coherent.

AA-CBR maps the problem of determining the outcome for a new case into a membership problem within the grounded extension of an AA framework (AAF) [Dung, 1995] obtained from the case base CB, the new case $(N, ?)$ and the default outcome δ. In general, an *AAF* is a pair $(Args, \rightsquigarrow)$, where $Args$ is a set (of *arguments*) and $\rightsquigarrow$ is a binary relation on $Args$ (where, for $\mathsf{a}, \mathsf{b} \in Args$, if $\mathsf{a} \rightsquigarrow \mathsf{b}$, then we say that a *attacks* b and that a is an *attacker of* b). For a set of arguments $E \subseteq Args$ and an argument $\mathsf{a} \in Args$, E *defends* a if for all $\mathsf{b} \rightsquigarrow \mathsf{a}$ there exists $\mathsf{c} \in E$ such that $\mathsf{c} \rightsquigarrow \mathsf{b}$. Then, the *grounded extension* of $(Args, \rightsquigarrow)$ can be constructed as $\mathbb{G} = \bigcup_{i \geqslant 0} G_i$, where G_0 is the set of all unattacked arguments, and $\forall i \geqslant 0$, G_{i+1} is the set of arguments that G_i defends. For any $(Args, \rightsquigarrow)$, the grounded extension $\mathbb{G}$ always exists and is unique.

AA-CBR encompasses the following two modules, respectively (1) extracting an AAF from a coherent case base and a default outcome, and (2) determining the outcome of a new case from the grounded extension of the AAF augmented to take into account the new case.

Module 1 – $aaf(CB, \delta)$ gives $(Args, \rightsquigarrow)$ with:

- $Args = CB \cup \{(\{\}, \delta)\}$;
- for $(X, o_X), (Y, o_Y) \in CB \cup \{(\{\}, \delta)\}$, it holds that $(X, o_X) \rightsquigarrow (Y, o_Y)$ iff
  1. $o_X \neq o_Y$, and (different outcomes)
  2. $Y \subsetneq X$, and (specificity)
  3. $\nexists (Z, o_X) \in CB$ with $Y \subsetneq Z \subsetneq X$. (concision)

The *default argument* $(\{\}, \delta)$ represents the outcome obtained in the absence of any information (i.e. features).

Module 2 – $outcome((Args, \rightsquigarrow), N)$ gives *the AA-CBR outcome* of $(N, ?)$ defined as follows:

i. let $(Args_N, \rightsquigarrow_N)$ be the AAF with
   - $Args_N = Args \cup \{(N, ?)\}$;
   - $\rightsquigarrow_N = \rightsquigarrow \cup \{((N, ?), (Y, o_Y)) : (Y, o_Y) \in Args, Y \not\subseteq N\}$, i.e. $(Args_N, \rightsquigarrow_N)$ extends $(Args, \rightsquigarrow)$ with the new case argument $(N, ?)$ that attacks all arguments with 'irrelevant' features, i.e. features not in N;

ii. let $\mathbb{G}$ be the grounded extension of $(Args_N, \rightsquigarrow_N)$; $\mathbb{G}$ is non-empty, as $(N, ?)$ is unattacked;

iii. then the *AA-CBR outcome* of $(N, ?)$ is
   - δ, if $(\{\}, \delta) \in \mathbb{G}$;
   - $\overline{\delta}$, otherwise, if $(\{\}, \delta) \notin \mathbb{G}$.

Example 2.2. Given CB, δ and $(N, ?)$ from Example 2.1, $outcome(aaf(CB, \delta), N)$ gives $\overline{\delta}$. Indeed $(Args, \rightsquigarrow) = aaf(CB, \delta)$ is the AAF depicted in Figure 1 when the polygon node and all arrows from it are ignored, and $(Args_N, \rightsquigarrow_N)$ from step (i) in Module 2 is the full AAF in the figure.[1] The grounded extension of $(Args_N, \rightsquigarrow_N)$ is

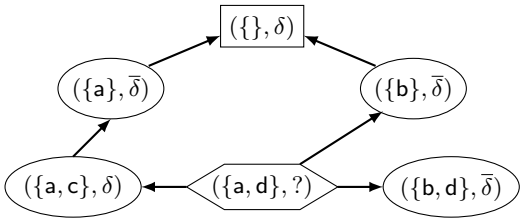


Figure 1: $(Args_N, \rightsquigarrow_N)$ for Example 2.2.

$\mathbb{G} = \{(\{\mathsf{a}, \mathsf{d}\}, ?), (\{\mathsf{a}\}, \overline{\delta})\}$, and $(\{\}, \delta) \notin \mathbb{G}$. Note that, in $(Args_N, \rightsquigarrow_N)$, $(\{\mathsf{b}, \mathsf{d}\}, \overline{\delta}) \not\rightsquigarrow (\{\}, \delta)$ because of the concision requirement: $(\{\mathsf{b}\}, \overline{\delta})$ is a more concise attacker of $(\{\}, \delta)$. Note also that $(N, ?)$ attacks all cases that have features not present in N.

For our purposes, a **logic program** is a set $\mathcal{P}$ of ground rules of the form $h \leftarrow p_1, \ldots, p_s, \mathsf{not}\ p_{s+1}, \ldots, \mathsf{not}\ p_{s+t}$. where $s, t \geq 0$, h and each p_i are atoms, and not is negation as failure. h is called the head, and $p_1, \ldots, p_s, \mathsf{not}\ p_{s+1}, \ldots, \mathsf{not}\ p_{s+t}$ the body of such a rule. A fact is a rule $h \leftarrow .$ with an empty body. All logic programs in this paper are stratified, and can be ascribed a meaning using the perfect model semantics[Apt and Bol, 1994]. We say that an atom a is *provable* (in $\mathcal{P}$) if $a \in model(\mathcal{P})$.

3 ANNA METHODOLOGY – PART I

In this section we describe our Artificial Neural Networks with Argumentation methodology for prediction (ANNA in short), summarised in Figure 2, but ignoring rule extraction and rule-based prediction (part II) until Section 5.

3.1 AUTOENCODER + FEATURE SELECTION

ANNA assumes the availability of a *training dataset* consisting of a (possibly large) set of examples, (referred to as $\mathcal{L}$ in the remainder), each amounting to a set of *features* from a

[1] Here, as conventional, the AAF is depicted as a graph with nodes holding arguments and arrows indicating attacks. We use different shapes for nodes for readability: the default argument $(\{\}, \delta)$ and the new case $(\{\mathsf{a}, \mathsf{d}\}, ?)$ argument are enclosed in rectangle and polygon respectively, while other arguments are enclosed in ellipses.

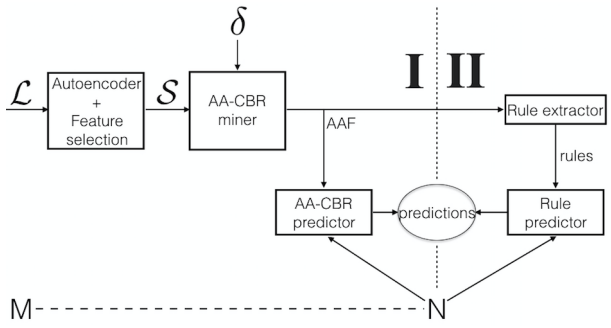


Figure 2: A bird's eye view of ANNA. **Part I**: during training with dataset $\mathcal{L}$ a trimmed dataset $\mathcal{S}$ is obtained, that, together with default outcome δ, gives rise to an AAF used for predicting the outcome for (any) M via its trimmed version N. **Part II**: the AAF is mapped onto rules for making equivalent predictions for M via N.

given set $\mathbb{F}_{\mathcal{L}}$ and an *outcome* from a set $\mathbb{O}$. We assume that there are exactly two outcomes in $\mathbb{O}$, i.e. prediction is a binary classification problem. The set $\mathbb{F}_{\mathcal{L}}$ may be large, especially if features are assignments of alternative values to attributes (e.g. 'colour' may take many different values). ANNA relies upon an autoencoder for capturing the most salient features from the training dataset, resulting in a *trimmed dataset* (referred to as $\mathcal{S}$ in the remainder), which may be much smaller than the original $\mathcal{L}$ and consisting of examples with much fewer features (from a trimmed set $\mathbb{F} \subseteq \mathbb{F}_{\mathcal{L}}$). Formally, $\mathcal{S} = \{(Y, o) : (X, o) \in \mathcal{L}, Y = X \cap \mathbb{F}\}$. ANNA enforces that feature selection, leading to $\mathbb{F}$, is such that $\mathcal{S}$ is *coherent* and thus representative of the original $\mathcal{L}$ (if coherent to start with) and devoid of noise (if $\mathcal{L}$ was not coherent). Thus feature selection guarantees that $\mathcal{S}$ is "rational".

Example 3.1. Let examples in $\mathcal{L}$ be characterised by 5 attributes $a_1, \ldots, a_5$ and an outcome amongst o_1, o_2. Suppose each attribute may take one of 4 distinct, discrete values, say $v^1_{a_i}, \ldots, v^4_{a_i}$ for a_i. Then $\mathbb{F}_{\mathcal{L}} = \{a_1 = v^1_{a_1}, \ldots, a_5 = v^4_{a_5}\}$ consists of 20 features (binary attribute-value pairs). Suppose that ANNA is realised so as to select 4 features, namely $\mathbb{F}$ consists of 4 elements, e.g. $\mathbb{F} = \{a_1 = v^1_{a_1}, a_2 = v^4_{a_2}, a_3 = v^2_{a_3}, a_4 = v^4_{a_4}\} = \{\mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{d}\}$: if $\mathcal{L} = \{(\{\mathsf{a}, a_1 = v^2_{a_1}, a_5 = v^1_{a_5}\}, o_1), (\{\mathsf{a}, a_1 = v^3_{a_1}\}, o_1)\}$ then $\mathcal{S} = \{(\{\mathsf{a}\}, o_1)\}$. Note that if $\mathcal{L}$ had also included $(\{\mathsf{a}, a_5 = v^1_{a_5}\}, o_2)$, then ANNA would not have selected $\mathbb{F}$, as this would have resulted in an incoherent $\mathcal{S} = \{(\{\mathsf{a}\}, o_1), (\{\mathsf{a}\}, o_2)\}$.

In our realisation of the ANNA methodology in this paper (and in particular in the experiments we present in Section 4) we use a simple autoencoder with one hidden layer as shown in Figure 3, with (for $X \subseteq \mathbb{F}_{\mathcal{L}}$): (i) an encoder function $f(X) = \sigma(XW^{(1)})$, and (ii) a decoder function $\sigma(f(X)W^{(2)})$, where $W^{(1)}, W^{(2)}$ are the weight parameters in the encoder and decoder, respectively.

To select $\mathbb{F}$, we average the weights in $W^{(1)}$ for each input and select the top F factors. F can be chosen in many alternative ways, e.g. iteratively (starting from a small number of features, until a coherent S is obtained) or empirically (as in the experiments in Section 4, where F=22, with $|\mathcal{L}| = 126$).

In the remainder we will refer to elements of $\mathcal{L}$ as *original examples*, and to elements of $\mathcal{S}$ simply as *examples*.

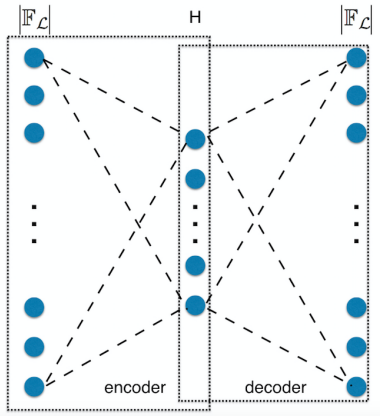


Figure 3: Autoencoder used in ANNA: $|\mathbb{F}_{\mathcal{L}}|$ binary features are used to train the autoencoder to obtain a **code** (hidden layer **h** of size $H < |\mathbb{F}_{\mathcal{L}}|$) that best captures the input $|\mathbb{F}_{\mathcal{L}}|$ features.

3.2 AA-CBR MINER

Note that $\mathcal{S}$ is a coherent case base, in the sense of Section 2, and examples in $\mathcal{S}$ can be seen as cases. ANNA assumes that a default δ can be naturally identified in $\mathbb{O}$, as the prediction that can be legitimately made in the absence of information about features. Then, the AA-CBR miner returns $aaf(\mathcal{S}, \delta)$ (see Module 1 in Section 2). Thus, if $\mathcal{S} = CB$ in Example 2.1, the AAF given by the AA-CBR miner is in Example 2.2.

Note that the AAF returned by the AA-CBR miner can be seen as a model of the trimmed dataset, and, if the feature selection is rational, also of the original training dataset. This model identified conflicts between (original) examples, that need to be resolved every time that a prediction is to be made.

3.3 AA-CBR PREDICTOR

The AA-CBR predictor takes a set M of features in $\mathbb{F}_{\mathcal{L}}$, characterising an unseen example with unknown outcome, and predicts an outcome from $\mathbb{O}$ as follows. Let the *trimming function* $\tau : \wp(\mathbb{F}_{\mathcal{L}}) \mapsto \wp(\mathbb{F})$ be $\tau(X) = X \cap \mathbb{F}$, for any $X \subseteq \mathbb{F}_{\mathcal{L}}$. Note that trimming different sets of features may result into the same set, e.g. in Example 3.1 $\tau(\{\mathsf{a}, a_1 = v^2_{a_1}, a_5 = v^1_{a_5}\}) = \tau(\{\mathsf{a}, a_1 = v^3_{a_1}\}) = \{\mathsf{a}\}$. The AA-CBR predictor takes $N = \tau(M)$ in input, alongside $(Args, \rightsquigarrow)$ given by the AA-CBR miner, and computes $outcome((Args, \rightsquigarrow), N)$ (see Module 2 in Section 2), as in Example 2.2.

4 EMPIRICAL EVALUATION

In this section we conduct experiments with a publicly available dataset, showing that ANNA is an effective method when compared to end-to-end ANNs and to a method based on decision trees. The latter is chosen for comparison as decision trees also produce logical rules for justifying predictions, as ANNA-Part II (see Section 5).

The dataset consists of 8124 examples of gilled mushrooms classified as edible or poisonous. Each example is characterised by 22 (categorical) attributes that can take a number of different values, leading to 126 (binary) features. In our experiments we choose several subsets of this dataset as $\mathcal{L}$, but, for all choices, $|\mathbb{F}_{\mathcal{L}}| = 126$.

In our experiments, $H \in \{22, 30\}$ (where H is the size of the hidden layer, see Figure 3), $|\mathbb{F}| = 22$, for $\mathbb{F}$ the trimmed set of features, $\mathbb{O} = \{\text{edible}, \text{poisonous}\}$ and $\delta = \text{edible}$. This means that in the absence of any information, i.e. features, about a mushroom, it can be deemed edible, as represented by $(\{\}, \delta)$. However, as soon as a mushroom with features is encountered, it being edible has to be justified by countering all the relevant examples of poisonous mushrooms.

We use sigmoid as activation function in the autoencoder and binary cross entropy as loss function. We have experimented with $|\mathbb{F}| \in \{22, 30, 50\}$, with tanh and ReLU as activations functions and with various optimizers, but report results for the best performing combinations of parameters.

Table 1: 5-fold cross validation results

Hidden layer size 22	**Precision**	**Recall**	$\mathbf{F_1}$
ANNA	**0.97**	**0.96**	**0.958**
Autoencoder + ANN	0.938	0.894	0.878
ANN	0.934	0.888	0.87
Hidden layer size 30			
ANNA	**0.97**	**0.962**	**0.962**
Autoencoder + ANN	0.932	0.886	0.86
ANN	0.936	0.896	0.88

In Table 1 we report 5-fold cross validation results, using weighted averages for each metric, for a stand-alone ANN, for a combination of Autoencoder+ANN, and for ANNA. The chosen ANN has one hidden layer and uses sigmoid as activation function and softmax to make predictions. The hyper-parameters were optimised using the Adam method [Kingma and Ba, 2014] with learning rate 0.001. For Autoencoder+ANN, we use the learnt weights $W^{(1)}$ from the encoder, which we do not optimise during training, and softmax for classification. In both cases, we trained for 50 epochs or until the performance on the development set stopped improving. In the case of ANNA, we use the learnt weights from the encoder to select the top 22 features. As shown in Table 1, ANNA performs better than the two ANN approaches with differences in F_1 up to 8% and 10% when using a size 22 and 30 hidden layer, respectively.

We also conducted experiments to test whether ANNA can better cope with smaller datasets than the Autoencoder+ANN method (arguably the better performing of the two end-to-end ANN methods). Hence we run experiments on 6000 randomly drawn examples and 5000 randomly drawn examples, respectively, and tested on the remaining examples in the starting dataset. We repeated the experiments 5 times and report the average performances in Table 2. Here as well we use the learnt weights $W^{(1)}$ from the encoder in Autoencoder+ANN and softmax for prediction. In this set of experiments, we also used for comparison a method, referred to simply as Decision Tree (DT) in Table 2, that also uses an autoencoder for feature selection. For ANNA and DT alike, we use the learnt weights from the encoder to select the top 22 features as $\mathbb{F}$ and give $\mathcal{S}$ as discussed in Section 3.1. We used information gain for DT and selected 1 as the minimum number of samples required to be at a leaf node to mimic ANNA treating each example as an argument.

Table 2: Average of 5 runs of training on a reduced dataset and testing on the remaining examples.

Training set size: 6000, Testing set size: 2124			
Hidden layer size 22	**Precision**	**Recall**	$\mathbf{F_1}$
ANNA	**0.978**	**0.976**	**0.976**
Autoencoder + ANN	0.802	0.642	0.61
Decision Tree (DT)	0.858	0.774	0.762
Hidden layer size 30			
ANNA	**0.966**	**0.964**	**0.966**
Autoencoder + ANN	0.802	0.638	0.604
Decision Tree (DT)	0.852	0.772	0.766
Training set size: 5000, Testing set size: 3124			
Hidden layer size 22	**Precision**	**Recall**	$\mathbf{F_1}$
ANNA	**0.954**	**0.954**	**0.954**
Autoencoder + ANN	0.84	0.76	0.75
Decision Tree (DT)	0.876	0.828	0.826
Hidden layer size 30			
ANNA	**0.97**	**0.97**	**0.97**
Autoencoder + ANN	0.84	0.756	0.748
Decision Tree (DT)	0.886	0.844	0.844

The experiments on reduced datasets show that Autoencoder+ANN is less performing than ANNA and DT. ANNA performs better than DT throughout all experiments, with improvements in F_1 up to 20% and 12% with training set size of 6000 and 5000 examples, respectively.

5 ANNA METHODOLOGY – PART II

In this section we describe how ANNA generates logic programs that capture reasoning with examples in $\mathcal{S}$. These rules allow to make exactly the same predictions for new examples as the AA-CBR predictor. At the same time, these rules summarise the structure of $\mathcal{S}$ and, by extension, that of $\mathcal{L}$. In particular, they concisely describe in logical terms what the prediction should be and which features should influence it. The rules are generic for $\mathcal{S}$ but when a prediction for a new example is made its features are added as facts to the rules.

We give ANNA Rule extractor/predictor (Figure 2) below.

5.1 RULE EXTRACTOR

Unless stated otherwise, let $(Args, \rightsquigarrow) = aaf(\mathcal{S}, \delta)$ be the AAF returned by the AA-CBR miner (see Section 3.2). Informally, rule extraction works as follows.

- Take $(\{\}, \delta) \in Args$ and its attackers (if any).
  - Create a rule stating that $(\{\}, \delta)$ is accepted unless any of its attackers are accepted.
  - For each attacker $(\{\mathsf{f}_1, \ldots, \mathsf{f}_m\}, \overline{\delta})$ of $(\{\}, \delta)$, create a rule stating that it is accepted if the features $\mathsf{f}_1, \ldots, \mathsf{f}_m$ hold, unless any of its attackers are accepted.
  - If there are no attackers, create a rule stating that the argument is accepted if all its features hold.
- Repeat for each attacker and its attackers in turn.

Formally, assume a naming function $name$ that assigns a unique name to every argument in $Args$: we write $C : \mathsf{a}$ to indicate that C names argument a, and we name $(\{\}, \delta)$ by def. Define the procedure $extract(C, (Args, \rightsquigarrow), \mathcal{R})$, taking the name C of an argument in $Args$ and a rule set $\mathcal{R}$, thus:

1. Given $C : (\{\mathsf{f}_1, \ldots, \mathsf{f}_m\}, o)$, let $\{C_1, \ldots, C_k\}$ be the set of the names of all the attackers of C in $(Args, \rightsquigarrow)$.
2. Add the following rule to $\mathcal{R}$, to obtain $\mathcal{R}'$:
   $acc(C) \leftarrow \mathsf{f}_1, \ldots, \mathsf{f}_m, \mathsf{not}\ acc(C_1), \ldots, \mathsf{not}\ acc(C_k)$.
3. If $\{C_1, \ldots, C_k\} = \{\}$ then return $\mathcal{R}'$.
4. For every $C_i \in \{C_1, \ldots, C_k\}$, compute $extract(C_i, (Args, \rightsquigarrow), \mathcal{R}')$, to obtain $\mathcal{R}'_1, \ldots, \mathcal{R}'_k$.
5. Return $\mathcal{R}'_1 \cup \ldots \cup \mathcal{R}'_k$.

The Rule extractor computes $extract(def, (Args, \rightsquigarrow), \{\})$, whose output is a logic program, henceforth called $\mathcal{P}$.

Example 5.1. Consider $(Args, \rightsquigarrow)$ from Example 2.1, depicted as in Figure 1 but without the polygon and attacks from it. Let $def, C_\mathsf{a}, C_\mathsf{b}, C_\mathsf{c}, C_\mathsf{d}$ be the names of arguments $(\{\}, \delta), (\{\mathsf{a}\}, \overline{\delta}), (\{\mathsf{b}\}, \overline{\delta}), (\{\mathsf{a}, \mathsf{c}\}, \delta), (\{\mathsf{b}, \mathsf{d}\}, \overline{\delta})$, respectively. Then $\mathcal{P}$ consists of the following rules:

$acc(def) \leftarrow \mathsf{not}\ acc(C_\mathsf{a}), \mathsf{not}\ acc(C_\mathsf{b})$.
$acc(C_\mathsf{a}) \leftarrow \mathsf{a}, \mathsf{not}\ acc(C_\mathsf{c})$.
$acc(C_\mathsf{c}) \leftarrow \mathsf{a}, \mathsf{c}$.
$acc(C_\mathsf{b}) \leftarrow \mathsf{b}$.

These rules describe how $acc(def)$ can be proved. Note that there is no rule for acceptance of C_d, because C_d does not attack any other argument. In particular, there is no (directed) path from it to $(\{\}, \delta)$ in $(Args, \rightsquigarrow)$. Hence, C_d is never reached within $extract(def, (Args, \rightsquigarrow), \{\})$. This illustrates that the feature d is not important in proving $acc(def)$.

Note that by construction $\mathcal{P}$ is stratified as $(Args, \rightsquigarrow)$ has no cycles and every $\mathsf{not}\ acc(C')$ in the body of rules with head $acc(C)$ refers to some child (attacker) C' of C.

5.2 RULE PREDICTOR

The generic rules forming $\mathcal{P}$ can be used to predict the outcome of new example $N = \tau(M)$ by adding each feature from N as a fact to give $program(\mathcal{P}, N) = \mathcal{P} \cup \{\mathsf{f} \leftarrow . : \mathsf{f} \in N\}$. Henceforth, $\mathcal{P}_N$ denotes the logic program $program(\mathcal{P}, N)$.

Example 5.2. In Example 5.1, $\mathcal{P}_N$ consists of rules:

$acc(def) \leftarrow \mathsf{not}\ acc(C_\mathsf{a}), \mathsf{not}\ acc(C_\mathsf{b})$.
$acc(C_\mathsf{a}) \leftarrow \mathsf{a}, \mathsf{not}\ acc(C_\mathsf{c})$.
$acc(C_\mathsf{c}) \leftarrow \mathsf{a}, \mathsf{c}$.
$acc(C_\mathsf{b}) \leftarrow \mathsf{b}$.
$\mathsf{a} \leftarrow .$
$\mathsf{d} \leftarrow .$

Note: as $\mathcal{P}$ is stratified, $\mathcal{P}_N$ is stratified. The Rule predictor takes $(N, ?)$ and $\mathcal{P}_N$ in input, and predicts δ if $acc(def)$ is provable, and $\overline{\delta}$ otherwise. With $\mathcal{P}_N$ from Example 5.2, $model(\mathcal{P}_N) = \{\mathsf{a}, \mathsf{d}, acc(C_\mathsf{a})\}$, so $acc(def) \notin model(\mathcal{P}_N)$, i.e. $acc(def)$ is not provable, and the prediction is $\overline{\delta}$.

The Rule predictor and the AA-CBR predictor are equivalent in the following sense:

Theorem 5.1. *Let $\mathcal{S}$, δ and N be given. Let $(Args, \rightsquigarrow) = aaf(\mathcal{S}, \delta)$, $\mathcal{P} = extract(def, (Args, \rightsquigarrow), \{\})$, and $\mathcal{P}_N = program(\mathcal{P}, N)$. Then the AA-CBR outcome of $(N, ?)$ is δ iff $acc(def) \in model(\mathcal{P}_N)$.*

Proof. First note that by construction $\mathcal{P}_N$ can be stratified while traversing $(Args, \rightsquigarrow)$ from $(\{\}, \delta)$ to the leaves, by putting the rule for acceptance of $(\{\}, \delta)$ on the top stratum, and the rule for acceptance of any other argument on the stratum lower by the length of the longest (directed) path from that argument to $(\{\}, \delta)$. The facts from N can then be put on the lowest stratum.

Given such a partition, the iterated fixpoint construction effectively mimics the acceptance of $(\{\}, \delta)$ in the grounded extension of $(Args_N, \rightsquigarrow_N)$ given by $outcome((Args, \rightsquigarrow), N)$ (see Module 2 in Section 2). First, for any rule $acc(A) \leftarrow \ldots, p, \ldots.$ with $p \notin N$, $acc(A)$ is not provable. Accordingly, $A : (X, o)$ is attacked by $(N, ?)$, so $(X, o) \notin \mathbb{G}$. Then, for any $acc(C) \leftarrow p_0, \ldots, p_s.$, $acc(C)$ is provable. Accordingly, $acc(C)$ names a leaf in $(Args_N, \rightsquigarrow_N)$ and so is in $\mathbb{G}$. Any argument A attacked by C is thus not in $\mathbb{G}$. Accordingly, any rule with head $acc(A)$ has in its body some $\mathsf{not}\ p$ with p provable, so $acc(A)$ is not provable.

This iteration over rules finishes with the rule $acc(def) \leftarrow \mathsf{not}\ acc(C_1), \ldots, \mathsf{not}\ acc(C_k)$, whence $acc(def)$ is provable only if none of $acc(C_i)$ is. By the reasoning above, this amounts to $\mathbb{G}$ defending $(\{\}, \delta)$. In particular, $(\{\}, \delta) \in \mathbb{G}$ iff $acc(def) \in model(\mathcal{P}_N)$. □

This result shows that in ANNA, both AA-CBR predictor and Rule predictor can be equivalently used to make predictions for new examples. Thus, the two predictors have the same predictive performances (see Section 4).

6 ILLUSTRATING EXPLANATIONS

In this section we illustrate the explanatory power of ANNA with the mushroom dataset used in Section 4.

6.1 LOGICAL EXPLANATIONS

In what follows, we use the logic program $\mathcal{P}$ obtained for $\mathcal{S}$ from one of the datasets from one of the experiments (with $|\mathcal{L}| = 5000$) in Section 4. This $\mathcal{S}$ consists of 306 examples, and $\mathcal{P}$ consists of 13 rules. The rule in $\mathcal{P}$ concerning the acceptance of $(\{\}, \delta)$ is

$$acc(def) \leftarrow \mathsf{not}\ acc(131^{def}), \ldots, \mathsf{not}\ acc(34^{def}). \quad (1)$$

with 7 elements in the body (of which we spell out 2).[2] This rule says that the outcome is δ—namely, a mushroom being edible—unless at least one of the seven exceptions applies. Let us look at one particular exception, induced by example 131. The rule concerning acceptance of 131 is

$$acc(131^{def}) \leftarrow \mathsf{f}_1, \ldots, \mathsf{f}_{10}, \mathsf{not}\ acc(2504^{131}). \quad (2)$$

with 11 elements in the body, including one exception. This rule says that given features $\mathsf{f}_1, \ldots, \mathsf{f}_{10}$, example 131 is an exception to the default outcome, but that it also admits an exception—namely example 2504.

The exception to $acc(131^{def})$ is given by the rule

$$acc(2504^{131}) \leftarrow \mathsf{gill-size\_broad}. \quad (3)$$

This rule says that an exception to example 131 is obtained as soon as the feature $\mathsf{gill-size\_broad}$ is present. Note that example 2504 has no exceptions.

Suppose that $N = \{\mathsf{f}_1, \ldots, \mathsf{f}_{10}, \mathsf{gill-size\_broad}\}$ and that only exception 131 to rule (1) applies in $\mathcal{P}_N$. Then rules (1), (2), (3) and the facts obtained from N form the basis for a logical explanation for why δ is predicted for N.

[2] Examples in $\mathcal{S}$ are named by integer numbers.

6.2 DIALECTICAL EXPLANATIONS

The prediction for the outcome of a given N in the context of some $\mathcal{S}$ can be explained *dialectically* too, by using $(Args, \rightsquigarrow) = aaf(\mathcal{S}, \delta)$ and $(Args_N, \rightsquigarrow_N)$ given by $outcome((Args, \rightsquigarrow), N)$ (see Section 3). To this end, imagine a debate between a proponent, P, seeking to establish that a given mushroom is edible, and an opponent, O, seeking to establish that it is poisonous. This debate can be visualised via $(Args_N, \rightsquigarrow_N)$, with arguments labelled as P and O. The debate would unfold as follows.

Given some particular mushroom represented by $(N, ?)$, P moves argument $(\{\}, \delta)$, representing that by default a mushroom is edible. Note that $(\{\}, \delta)$ has no features, i.e. does not describe any mushroom in particular, but rather represents a generic mushroom. So accepting $(\{\}, \delta)$ means that in the absence of any information about a given mushroom, it should be edible. O then argues for the mushroom in question to be poisonous by putting forward examples of poisonous mushrooms. To this end, the opponent puts forward arguments with the non-default outcome $\overline{\delta}$. For instance, suppose that, mirroring Section 6.1, O uses the argument named 131.

P then has to counter-argue O's argument(s). P has two ways to do that. i) Either the example put forward by O is of a poisonous mushroom that has features different from N. For instance, it may be that $\mathsf{f}_{10} \notin N$. If this happens, P argues that the example is 'irrelevant', whence $(N, ?)$ attacks argument 131. Generally, putting forward 'irrelevant' examples is useless. ii) Otherwise, P can find a more specific example of an edible mushroom that has all the features of the given example and, in addition, some more features that are still present in N. For instance, if $\mathsf{gill-size\_broad} \in N$, then P can use argument 2504, which attacks argument 131.

The process continues in this way whereby O puts forward counter-examples and P tries to defend against those. If in the end O has no arguments to put forward and all of O's arguments are attacked by P's arguments, i.e. all leaves are labelled P, then P has established the default outcome, namely, that the mushroom in question is edible. Otherwise, if there is an applicable counter-example put forward by O that P cannot argue against, the default outcome cannot be upheld, and so the mushroom is deemed poisonous. For instance, suppose in the above that $(N, ?)$ attacks all the arguments attacking $(\{\}, \delta)$ but does not attack the one named 131, and suppose $\{\mathsf{f}_1, \ldots, \mathsf{f}_{10}, \mathsf{gill-size\_broad}\} \subseteq N$. Then P's last argument 2504, which is a leaf and hence unattacked, together with $(N, ?)$ defends $(\{\}, \delta)$, and so P wins the debate and establishes that the mushroom represented by N is edible.

7 RELATED WORK

Several works have integrated argumentation and machine learning (see e.g. [Cocarascu and Toni, 2016]). Perhaps the most relevant work to ours is that of [Amgoud and Serrurier, 2007] where argumentation and concept learning are used to reason about classifications. Arguments are examples and hypotheses. Preferences (assumed to be given) over arguments are employed to resolve conflicts stemming from an inconsistent training set, using the grounded extension to capture the version space model, while also supporting generation of argumentative explanations for classifications. In contrast, we advance a novel model using only examples. Our training sets are coherent (by construction) and the definition of attack incorporates a form of preference over arguments. Also, we provide dialectical and logical explanations, in terms of arguments and rules, respectively.

[d'Avila Garcez *et al.*, 2005] propose a Neural Argumentation Algorithm that translates argumentation frameworks to ANNs and affords semantic correspondence between the two. [Makiguchi and Sawamura, 2007] further this research to provide symbolic dialogues from ANNs. These works are not concerned with predictions or subsequent explanation of predictions. In [Thimm and Kersting, 2017] a proposal is formulated to extract (possibly inconsistent) rules from a dataset using machine learning, and then to use those rules in structured argumentation formalisms (see e.g. [Besnard *et al.*, 2014]) to classify new examples. We instead use ANNs to mine argumentation frameworks for reasoning and rule extraction.

In [Kontschieder *et al.*, 2016; Zhou and Feng, 2017; Balestriero, 2017] decision trees and ANNs are combined to give Neural Decision Trees. Such approaches can potentially be interpretable as trees are easier to analyse than ANNs and can cope with small datasets [Zhou and Feng, 2017]. By contrast, ANNA's data models are logic-based rather than probabilistic, and can be compactly represented via both (argumentation) graphs and (logic programming) rules.

Several approaches to explanation in AI have been proposed, e.g. see [Ribeiro *et al.*, 2018]. Rather than explaining existing methods, we have defined a novel method that is explainable, in two alternative ways (logically and dialectically), and in such a way that it can explain both predictions and underlying learnt model.

8 CONCLUSION AND FUTURE WORK

Our main contributions in this paper are as follows.

- Combining (a form of) Artificial Neural Networks (ANNs) and (a form of) Argumentation for solving binary classification problems, whereby ANNs perform feature selection and Argumentation performs classification.
- The resulting methodology, ANNA, provides argumentation-based predictions (classification) as well as logical rules that give equivalent predictions.
- ANNA performs well in our experiments.
- ANNA's predictions are explainable both dialectically and logically, and the explanations form argumentation-based and rule-based models of data.

In the future we plan to do (at least) the following: (i) further experiment with ANNA, considering other datasets and other forms of ANNs for feature selection or engineering; (ii) relax the notion of coherent case bases to deal with noise; (iii) incorporate probabilities over features; (iv) explore the application of logic programming transformation (e.g. see [Proietti and Pettorossi, 1995]) to simplify rule-based predictions; (v) compare our rule extraction to other approaches that mine rules from data, such as Inductive Logic Programming (e.g. see [Muggleton, 1991]); (vi) compare the forms of explanation we generate with those of other methods, e.g. that of [Ribeiro *et al.*, 2018].

References

[Amgoud and Serrurier, 2007] Leila Amgoud and Mathieu Serrurier. Agents that argue and explain classifications. *Autonomous Agents and Multi-Agent Systems*, 16(2):187–209, 2007.

[Andrews *et al.*, 1995] Robert Andrews, Joachim Diederich, and Alan B. Tickle. Survey and critique of techniques for extracting rules from trained artificial neural networks. *Know.-Based Syst.*, 8(6):373–389, December 1995.

[Apt and Bol, 1994] Krzysztof R Apt and Roland N Bol. Logic programming and negation: A survey. *The Journal of Logic Programming*, 19/20:9–71, 1994.

[Atkinson *et al.*, 2017] Katie Atkinson, Pietro Baroni, Massimiliano Giacomin, Anthony Hunter, Henry Prakken, Chris Reed, Guillermo Ricardo Simari, Matthias Thimm, and Serena Villata. Towards artificial argumentation. *AI Magazine*, 38(3):25–36, 2017.

[Balestriero, 2017] Randall Balestriero. Neural decision trees. *CoRR*, abs/1702.07360, 2017.

[Besnard *et al.*, 2014] Philippe Besnard, Alejandro Javier García, Anthony Hunter, Sanjay Modgil, Henry Prakken, Guillermo Ricardo Simari, and Francesca Toni. Introduction to Structured Argumentation. *Argument & Computation*, 5(1):1–4, 2014.

[Cocarascu and Toni, 2016] Oana Cocarascu and Francesca Toni. Argumentation for machine learning: A survey. In *6th International Conference on Computational Models of Argument*, pages 219–230. IOS Press, 2016.

[Čyras *et al.*, 2016a] Kristijonas Čyras, Ken Satoh, and Francesca Toni. Abstract Argumentation for Case-Based Reasoning. In *Principles of Knowledge Representation and Reasoning, 15th International Conference*, pages 549–552. AAAI Press, 2016.

[Čyras *et al.*, 2016b] Kristijonas Čyras, Ken Satoh, and Francesca Toni. Explanation for Case-Based Reasoning via Abstract Argumentation. In *6th International Conference on Computational Models of Argument*, pages 243–254. IOS Press, 2016.

[d'Avila Garcez *et al.*, 2005] Artur S. d'Avila Garcez, Dov M. Gabbay, and Luís C. Lamb. Value-based argumentation frameworks as neural-symbolic learning systems. *Journal of Logic and Computation*, 15(6):1041–1058, 2005.

[Dheeru and Karra Taniskidou, 2017] Dua Dheeru and Efi Karra Taniskidou. UCI machine learning repository, 2017.

[Dung, 1995] Phan Minh Dung. On the Acceptability of Arguments and its Fundamental Role in Nonmonotonic Reasoning, Logic Programming and n-person Games. *Artificial Intelligence*, 77:321–357, 1995.

[Erhan *et al.*, 2010] Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, 11:625–660, 2010.

[Fan and Toni, 2015] Xiuyi Fan and Francesca Toni. On computing explanations in argumentation. In *Proceedings of AAAI*, pages 1496–1502, 2015.

[Gasca *et al.*, 2006] Eduardo Gasca, José Salvador Sánchez, and R. Alonso. Eliminating redundancy and irrelevance using a new MLP-based feature selection method. *Pattern Recognition*, 39(2):313–315, 2006.

[Han *et al.*, 2017] Kai Han, Chao Li, and Xin Shi. Autoencoder feature selector. *CoRR*, abs/1710.08310, 2017.

[Hinton and Salakhutdinov, 2006] Geoffrey Hinton and Ruslan Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504 – 507, 2006.

[Kingma and Ba, 2014] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[Kontschieder *et al.*, 2016] Peter Kontschieder, Madalina Fiterau, Antonio Criminisi, and Samuel Rota Bulò. Deep neural decision forests. In *Proceedings of IJCAI*, pages 4190–4194, 2016.

[LeCun *et al.*, 2015] Yann LeCun, Yoshua Bengio, and Geoffrey E. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[Makiguchi and Sawamura, 2007] Wataru Makiguchi and Hajime Sawamura. A hybrid argumentation of symbolic and neural net argumentation (part II). In *Argumentation in Multi-Agent Systems, 4th International Workshop*, pages 216–233, 2007.

[Muggleton, 1991] Stephen Muggleton. Inductive logic programming. *New Generation Computing*, 8(4):295–318, 1991.

[Proietti and Pettorossi, 1995] Maurizio Proietti and Alberto Pettorossi. Unfolding - definition - folding, in this order, for avaoiding unnecessary variables in logic programs. *Theoretical Journal of Computer Science*, 142(1):89–124, 1995.

[Ribeiro *et al.*, 2018] Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. Anchors: High-precision model-agnostic explanations. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[Thimm and Kersting, 2017] Matthias Thimm and Kristian Kersting. Towards argumentation-based classification. In *Logical Foundations of Uncertainty and Machine Learning, Workshop at IJCAI'17*, 2017.

[Verikas and Bacauskiene, 2002] A. Verikas and M. Bacauskiene. Feature selection with neural networks. *Pattern Recogn. Lett.*, 23(11):1323–1335, 2002.

[Wang *et al.*, 2017] Shuyang Wang, Zhengming Ding, and Yun Fu. Feature selection guided auto-encoder. In *Proceedings of AAAI*, pages 2725–2731, 2017.

[Zhou and Feng, 2017] Zhi-Hua Zhou and Ji Feng. Deep forest: Towards an alternative to deep neural networks. In *Proceedings of IJCAI*, pages 3553–3559, 2017.

ScenarioNet: An Interpretable Data-Driven Model for Scene Understanding

Zachary A. Daniels[1], **Dimitris Metaxas**[1]
[1] Department of Computer Science, Rutgers University
zad7@cs.rutgers.edu, dnm@cs.rutgers.edu

Abstract

The ability for computational agents to reason about the high-level content of real world scene images is important for many applications. Existing attempts at complex scene understanding lack representational power, efficiency, and the ability to create robust meta-knowledge about scenes. We introduce **scenarios** as a new way of representing scenes. The scenario is an interpretable, low-dimensional, data-driven representation consisting of sets of frequently co-occurring objects that is useful for a wide range of scene understanding tasks. Scenarios are learned from data using a novel matrix factorization method which is integrated into a new neural network architecture, the **ScenarioNet**. Using ScenarioNet, we can recover semantic information about real world scene images at three levels of granularity: 1) scene categories, 2) scenarios, and 3) objects. Training a single ScenarioNet model enables us to perform scene classification, scenario recognition, multi-object recognition, content-based scene image retrieval, and content-based image comparison. ScenarioNet is **efficient because it requires significantly fewer parameters than other CNNs** while achieving similar performance on benchmark tasks, and it is **interpretable because it produces evidence in an understandable format** for every decision it makes. We validate the utility of scenarios and ScenarioNet on a diverse set of scene understanding tasks on several benchmark datasets.

1 Introduction

For many applications (e.g., robotics, human-machine teaming, surveillance, and autonomous vehicles), an agent must reason about the high-level content of real world scene images in order to make rational, grounded decisions that can be trusted by humans. It is often also necessary to have models that are able to be interpreted by humans in order to further encourage trust and allow humans to understand the failure modes of the autonomous agent. For example, if a self-driving car makes an error, it is important to know what caused the error to prevent future situations where similar errors might arise. Recently, a lot of progress has been made in constructing algorithms and systems that address fundamental scene understanding tasks such as scene classification, object detection, and semantic segmentation as well as more complex scene understanding tasks such as visual question-answering, automatic relationship extraction, scene graph generation, and learning how to visually reason about objects in simple scenes (e.g., [Johnson *et al.*, 2016]). While existing methods for solving such tasks are impressive, they often lack the interpretability and semantic grounding needed to make them trustworthy for safety-critical tasks and tasks involving human-machine teaming.

In this paper, we present a novel interpretable data-driven model for scene understanding. Explainable machine learning models rely on two properties: 1) features should be low-dimensional and human-interpretable and 2) models should be simple (with few parameters), easy for humans to inspect, and operate in a principled, well-understood way. We introduce a low-dimensional, semantically-grounded, object-based representation for scene understanding called the "scenario" which addresses the first property. We then show how scenarios can be used to make convolutional neural networks (CNNs) more transparent, thus addressing the second property.

We introduce **scenarios**, an interpretable, data-driven representation for scene understanding. Scenarios are based on *sets of frequently co-occurring objects*. Scenarios should satisfy a few key properties:

1. Scenarios are composed of one or more objects.
2. The same object can appear in multiple scenarios, and this should reflect the context in which the object appears, e.g., {keyboard, screen, mouse} and {remote control, screen, cable box} both contain the "screen" object, but in the first scenario, the screen is a computer monitor, and in the second scenario, it is a television screen.
3. Scenes can be decomposed as combinations of scenarios, e.g., a bathroom scene instance might decompose into: {shower, bathtub, shampoo} + {mirror, sink, toothbrush, toothpaste} + {toilet, toilet paper}.
4. Scenarios are flexible and robust to missing objects. A scenario can be present in a scene without all of its constituent objects being present.

We propose **Pseudo-Boolean Matrix Factorization (PBMF)** to identify scenarios from data. PBMF takes a binary *Object-Scene* matrix and decomposes it into 1) a dictionary matrix where each basis vector is a scenario and 2) an encoding matrix that expresses a scene instance as a combination of scenarios. We integrate PBMF into a novel convolutional neural network architecture (CNN), the **ScenarioNet**.

ScenarioNet replaces the final convolutional layers in standard CNNs with the **scenario block** (see Fig. 1) which consists of three parts: 1) global pooling layers that identify the parts of an image ScenarioNet attends to when recognizing whether each scenario is present in an image, 2) layers that use a PBMF-based loss function to learn a dictionary of scenarios and predict the presence and strength of each scenario for a given image, and 3) layers equivalent to a multinomial logistic regression model that use scenarios as low-dimensional features for predicting the scene category. During training, ScenarioNet only requires information about the *presence* (but not location) of objects in an image. For scene classification, class labels are also needed during training. During testing, only images are given.

Using ScenarioNet, we can recover semantic information about scene images at three levels of granularity: 1) scene categories, 2)

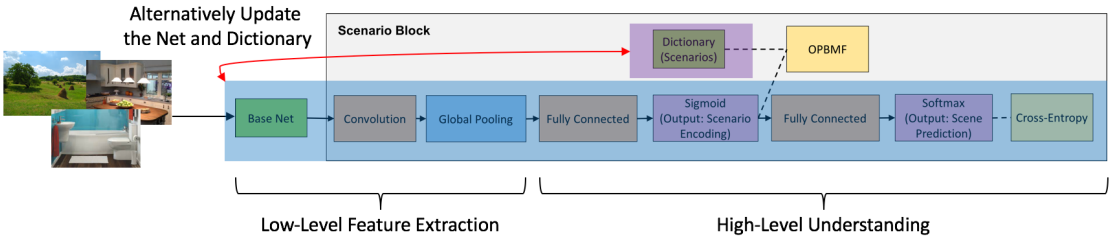


Figure 1: The **scenario block** replaces the final fully connected layers of a standard CNN and consists of: 1) global pooling layers that identify which parts of an image ScenarioNet attends to when recognizing whether a scenario is present in a given image, 2) layers that use a PBMF-based loss function to finetune a dictionary of scenarios and predict the presence of each scenario for a given image, and 3) layers equivalent to multinomial logistic regression that use scenarios as low-dimensional, interpretable features for scene classification.

scenarios, and 3) objects. This allows us to train a single ScenarioNet model capable of performing 1) scene classification, 2) scenario recognition, 3) multi-object recognition, 4) content-based scene image retrieval, and 5) content-based image comparison.

ScenarioNet has several advantages over other CNNs. It is computationally efficient because it requires significantly fewer parameters than other CNNs in order to achieve similar performance on benchmark tasks, and it is interpretable because it produces semantically- and visually-grounded evidence when making decisions. For example, for scene classification, predicted scenarios are used as low-dimensional semantic features; humans can verify the presence of each predicted scenario in an image by examining the scenario-localizing attention maps produced by the network; and humans can inspect how much influence each scenario exerts when assigning a class. This helps us to understand how a network arrives at specific decisions.

We evaluate the utility of ScenarioNet using the SUNRGBD [Song *et al.*, 2015], ADE20K [Zhou *et al.*, 2017b], and MIT 67 Indoor Scenes [Quattoni and Torralba, 2009] datasets. We perform quantitative experiments on multi-object recognition, scene classification, and content-based image retrieval. We also show examples demonstrating the interpretability and expressiveness of ScenarioNet.

2 Related Work

2.1 Learning Meaningful Groups of Objects

Discovering meaningful groups of objects is not a new idea. The simplest object-based representations are those that utilize pairwise co-occurrence relationships between objects (e.g., [Rabinovich *et al.*, 2007]). Scenarios go one step further by efficiently learning groups of objects of varying size. Many works focus on hierarchical models relating objects and scenes. [Feng and Bhanu, 2016] constructs a tree-based hierarchy of concepts based on object co-occurrence graphs. Objects sharing an ancestor node can be grouped into scene concepts, an idea similar to our scenarios. Several issues exist with using a tree structure for specifying scene concepts. To compute explicit scenarios, one must identify where to cut the tree. Additionally, while individual concepts can belong to multiple scene concepts by cutting the tree at different ancestor nodes, it becomes hard to properly place objects in the hierarchy that serve different functions within different groups, e.g., a screen with a keyboard and mouse is different from a screen with a cable box and remote. Our scenarios address these issues and provide additional information, e.g., how important each object is to a given scenario and how to decompose scene instances into combinations of scenarios. Other tree-based and hierarchical models for scene understanding exist. [Choi *et al.*, 2012] introduces a tree structure where nodes represent objects and latent variables and edges represent positive and negative correlations between nodes. These trees implicitly capture scenarios while our work learns explicit scenarios. [Fan *et al.*, 2008] exploit hierarchies of concepts to build ontologies for content-based image retrieval. [Lan *et al.*, 2013] investigate context at three levels: individual objects, parts of objects, and visual composites.

Other groups focus on using sets of objects to aid object detection. [Li *et al.*, 2012] discovers groups of objects of arbitrary size, model these groups using deformable parts models, and directly detects these groups in images. [Cinbis and Sclaroff, 2012] constructs classifiers that operate over sets of objects using object-object and object-scene relations to re-score and remove noisy detections. ScenarioNet differs from these methods because it jointly learns to group objects and coarsely localize them in images.

2.2 Explainable Models for Visual Recognition

The AI community has placed greater importance on learning explainable models and making complex models more interpretable. This is especially important for visual recognition tasks where many state-of-the-art models rely on deep neural networks. Several solutions have been proposed to solve this problem. [Ribeiro *et al.*, 2016] proposes a general framework for making complex models interpretable by looking at local linear approximations of the model's behaviour. Other works focus on generating visual explanations of CNN features, e.g., [Oquab *et al.*, 2015], [Zhou *et al.*, 2016], [Selvaraju *et al.*, 2017], and [Lengerich *et al.*, 2017], but these methods do not generate semantic explanations. [Hendricks *et al.*, 2016] trains a CNN to recognize objects and then trains an RNN to generate natural language explanations for the recognition decisions. They extend the system to work with visual question answering systems and also generate attention maps [Park *et al.*, 2016]. ScenarioNet generates less sophisticated, yet still human-interpretable semantic descriptions but doesn't require training language models which require large databases of image-caption pairs.

3 Proposed Method

3.1 Identifying Scenarios from Data: Pseudo-Boolean Matrix Factorization

We begin our discussion of the technical details of our model by asking: how do we identify which sets of objects naturally group together to form scenarios? We start with a training set of scene instances and a finite set of predetermined objects. We have ground-

truth annotations for the presence (or lack thereof) of every object in every scene instance given by either humans or object detectors. For each training instance, we create a vector of object presences where each element corresponds to a specific object, and the element is 1 if the object is present and 0 otherwise. We concatenate these vectors to form a matrix A where each row corresponds to a specific object and each column is a training instance. After specifying the number of desired scenarios k (which can be estimated from the data), we decompose A into two smaller approximately binary matrices: a dictionary matrix W representing a set of scenarios and an encoding matrix H that expresses scene instances as combinations of scenarios. Each column of W represents a single scenario and each row represents an object. If element W_{ij} is 0 or very small, object i is not present in scenario j. As W_{ij} approaches 1, object i exerts more influence on scenario j. Each column of H represents a specific scene instance and each row represents a specific scenario. If element H_{ij} is 0 or very small, then scenario i is not present in scene instance j. As H_{ij} approaches 1, scenario i exerts more influence on scene instance j.

Formulation of PBMF

We propose identifying scenarios using an approximation of Boolean matrix factorization (BMF) [Miettinen *et al.*, 2008]. In BMF, A, W, and H are binary matrices and the matrix multiplication is Boolean (denoted as $\circ$):

$$\min_{W,H} ||(A - W \circ H)||_1 \; s.t. \; W \in \{0,1\}, H \in \{0,1\} \tag{1}$$

BMF is well-suited for identifying scenarios from data because: 1) it efficiently compresses and preserves information using low-dimensional representations; 2) the basis vectors are easy to interpret; 3) it discovers meaningful interactions between objects; and 4) the encoding vectors are sparse, so each instance is expressed by a small subset of scenarios.

We use a gradient descent-based approach to solve the optimization problem. The formulation in Eq. 1 is not continuous, so we approximate Boolean matrix multiplication as $W \circ H \approx \min(WH, 1)$ and relax the constraints to lie in $[0, 1]$. Using $\min(WH, 1)$ results in cases where the gradient vanishes, so we further approximate $\min(WH, 1) \approx \min(WH, 1 + 0.01WH)$. Our basic **Pseudo-Boolean Matrix Factorization (PBMF)** formulation becomes:

$$\min_{W,H} ||(A - \min(WH, 1 + 0.01WH))||_F^2 \; s.t. \; W \in [0,1], H \in [0,1] \tag{2}$$

(Eq. 2) is still not perfectly suited for discovering scenarios. We add three additional terms: an orthogonality penalty to encourage diversity between scenarios and sparse penalties on the scenario dictionary and encoding to push W and H closer to binary matrices and improve interpretability. We introduce a weight matrix Ω that decreases the importance of common objects and increases the importance of rare objects during the factorization.

$$\begin{aligned} &\min_{W,H} ||\Omega \bullet (A - \min(WH, 1 + 0.01WH))||_F^2 \\ &+ \alpha_1 ||W^\intercal W - diag(W^\intercal W)||_F^2 + \alpha_2 ||W||_1 + \alpha_3 ||H||_1 \\ &s.t. \; W \in [0,1], H \in [0,1], \\ &\Omega_{ij} = \max\left(A_{ij} * \left(1 + \log\left(\frac{N_{instances}}{N_{objects}}\right)\right), 1\right) \end{aligned} \tag{3}$$

$\bullet$ denotes element-wise matrix multiplication. The αs represent tradeoff parameters.

3.2 ScenarioNet: Updating and Recognizing Scenarios from Visual Data

So far we've assumed we have perfect knowledge of all ground-truth object data. This means that if we're given a previously unseen scene instance, we can hold the scenario matrix constant and directly solve for the encoding matrix. In practice, we'll not have object data at test time. We need to learn how to recover the scenario encoding for a specific scene instance entirely from visual data. To do this, we integrate PBMF with CNNs. We propose **ScenarioNet**, a CNN that learns to identify and recognize scenarios from real-world visual data, performs scene classification using the predicted scenario encoding, and generates attention maps that highlight the regions the net focuses on when predicting whether a specific scenario is present in a given image. **ScenarioNet learns to predict an estimated scenario encoding matrix $\hat{H}$ and finetunes the dictionary W to adapt to the noisier $\hat{H}$. W also incorporates feedback from the scene classification task to improve discriminability**. The key architectural difference between ScenarioNet and other CNNs is the **scenario block** (see Fig. 1) which replaces the final fully connected layers used for classification in standard CNNs.

We now describe the rationale behind the scenario block. The final convolutional layers of a neural net such VGGNet are fed into a global average pooling (GAP) layer. This layer in combination with the class activation mapping technique [Zhou *et al.*, 2016] allows us to identify which parts of an image ScenarioNet attends to when determining if a scenario is present in the image. The output of the GAP layer is fed into a fully connected layer followed by a sigmoid transformation layer. The sigmoid layer outputs the **scenario encoding vector** and enforces each element of the vector is between 0 and 1. This vector tells us how present each scenario is in a given image. The scenario encoding layer feeds into a PBMF loss layer which finetunes the scenario dictionary and provides feedback to the network. The scenario encoding is also fed into a sequence of layers equivalent to a multinomial logistic regression model that uses scenarios as low-dimensional, interpretable features for scene classification.

Training ScenarioNet

During training, ScenarioNet only requires information about the *presence* (not location) of objects in an image. For scene classification, class labels are also needed during training. During testing, only images are given. First, the scenario dictionary is learned using ground-truth object presence data. Then, the net is trained to predict the scenario encodings while the dictionary is finetuned. Next, we train a softmax classifier for scene classification on top of a frozen net. Finally, we jointly finetune the net for scenario recognition and scene classification while once again finetuning the dictionary. It is useful to finetune only the last few layers of networks that have been previously trained for scene classification (e.g., on the Places dataset [Zhou *et al.*, 2017a]) since scenario recognition and scene classification are closely related. Each step of the finetuning process takes between 10 and 20 epoches. To finetune the dictionary while training the net, we use alternating projected gradient descent. During training, we hold the scenario dictionary constant and finetune the network using back-propagation in mini-batches to predict the encoding coefficients. After every four iterations, we hold the network constant and perform a full pass through the data to reconstruct $\hat{H}$ and finetune the scenario dictionary W using projected gradient descent. Alternatively, W can be efficiently finetuned using mini-batches by noting that the gradient of the PBMF loss w.r.t. W is able to be decomposed as a sum of gradients over sub-batches of $\hat{H}$; thus, we never have to compute the full $\hat{H}$ at any point in time.

Generating Evidence: Interpreting the Output of ScenarioNet

We now discuss how to interpret the output of ScenarioNet and by doing so understand how ScenarioNet makes interpretable decisions. Given an input image, ScenarioNet provides us with 1) a probabilistic scene class assignment, 2) a vector of scenario encoding coefficients, 3) the dictionary of scenarios, and 4) activation maps that can be used to localize the discriminative parts of each scenario. In Fig. 2, we show an example of decomposing a scene instance into its top-3 strongest detected scenarios using ScenarioNet. We see that ScenarioNet correctly predicts with high confidence that the scene category is "dining room". The top-3 scenarios support this: one focuses on dining areas, one on kitchen appliances, and one on decorative flowers. The encoding coefficient denotes the strength of each scenario. Note that all of the encoding coefficients are close to one since these are the strongest detected scenarios. As this coefficient decreases, the scenarios become less present. Encoding coefficients tend to cluster around 0 and 1. Recall that ScenarioNet uses scenarios as features for scene classification. We can define a scenario's *influence score* for a specific class to be the corresponding weight in the multinomial logistic regression model. If the influence is a large positive number, the scenario provides strong evidence for the specified class. If it is a large negative number, the scenario is strong evidence against a specific class. For this image, scenario 1 is very indicative of the scene class, while scenarios 2 and 3 are weakly indicative. We can also see how much influence each object exerts on each scenario. For example, in scenario 1, the "chandelier" and "chair" objects exert more influence when defining the scenario than the "buffet counter" object. By examining the scenario activation maps, we see that each predicted scenario is present and net attends to regions of the image containing objects present in the scenarios.

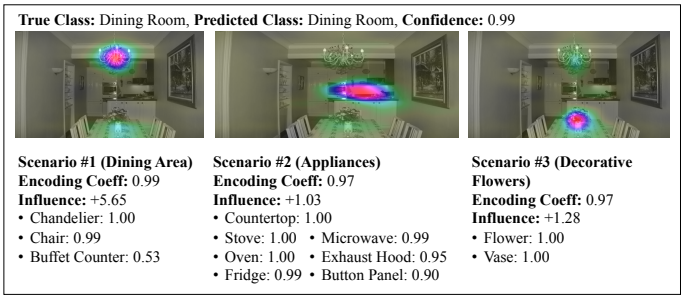


Figure 2: We demonstrate the explainability of ScenarioNet. We show the top-3 predicted scenarios for a dining room scene along with the corresponding activation maps. Please view in color.

Efficiency

Interestingly, our network has substantially fewer parameters than equivalent base architectures. For example, the final convolutional layers of VGG-16 typically consist of a 4096-by-4096 matrix followed by a 4096-by-#classes matrix for a total of $4096(4096 + \#classes)$ parameters. Our net uses a 512-by-#scenarios matrix followed by a #scenarios-by-#classes matrix for a total of $\#scenarios(512 + \#classes)$ parameters. Since #scenarios $<<$ 4096 (we use between $k = 25$ and $k = 70$ scenarios in our experiments), this results in over a 100x reduction in the number of parameters in the final layers, reduces the memory footprint of the *total* net by a factor of ˜10, and the net is ˜15% faster during testing.

4 Experimental Results and Analysis

In this section, we evaluate the reconstruction ability of PBMF, and also analyze the performance of ScenarioNet on three common scene understanding tasks: multi-object recognition, scene classification, and content-based scene image retrieval. We first explain the general experimental setup.

4.1 Experimental Setup

We conduct experiments on the SUNRGBD [Song *et al.*, 2015], ADE20K [Zhou *et al.*, 2017b], and MIT 67 Indoor Scenes [Quattoni and Torralba, 2009] datasets. We divide each dataset into separate training and test sets using the recommended splits for the SUNRGBD and MIT67 datasets and a random split for the ADE20K dataset. For each dataset, we only consider objects that appear in at least 1% of the training instances resulting in 55 objects for SUNRGBD, 193 for ADE20K, and 166 for MIT67. We use random cropping and horizontal mirroring to augment the training examples. For the SUNRGBD dataset, we use the 15 most frequently occurring scene classes, reserving 100 samples per class for test data, and generating 1000 samples per class for the training data. For the ADE20K dataset, we use the 31 most frequently occurring scene classes, reserving 25 samples per class for test data, and generating 500 samples per class for training data. For the MIT67 dataset, we use 67 scene classes, reserving 20 samples per class for test data, and generating 800 samples per class for training data. **We learn 25 scenarios for SUNRGBD, 70 for ADE20K, and 70 for MIT67. We use VGG-16 as our base CNN architecture**, replacing the final fully-connected layers with the scenario block. **For the MIT dataset, we only have object annotation data for about one-fifth of the training data, the amount of annotated data is very imbalanced between classes, and the annotations are much noisier than for the other datasets. These properties make learning scenarios on the MIT dataset much more difficult than for the other datasets, but we are still able to achieve relatively good results.** For this dataset, we learn the scenarios using the annotated portion of the training set and train a scene classifier on top of these scenarios for the full training set.

4.2 Reconstruction Error of PBMF

PBMF is a lossy factorization. We want to determine how much information about object presence is lost as a result of the decomposition. **For this experiment, we assume perfect, ground-truth knowledge of the object presences.** We consider three cases of PBMF: PBMF-Basic (Eq. 2), PBMF-Full (Eq. 3) with uniform weighting, and PBMF-Full using the proposed weight matrix. We compare to the SVD, NNSVD [Ding *et al.*, 2006], NMF [Paatero and Tapper, 1994], Greedy Boolean MF [Miettinen *et al.*, 2008], and Binary MF [Zhang *et al.*, 2007] as well as all-zeros and all-mean values baselines. We initialize the basis and encoding matrices using a procedure similar to [Zhang *et al.*, 2007]. Results are plotted in Fig. 3. PBMF-Basic works exceptionally well for reconstruction, generally losing to the much less constrained SVD. However, if we only focus on optimizing reconstruction error, we will overly prioritize common objects and might learn bases that lack diversity. As Fig. 3 demonstrates, adding orthogonality constraints and reweighing rare classes impacts the reconstruction error; however, we found adding these constraints results in dictionaries that are better suited for higher-level tasks such as scene classification and image retrieval.

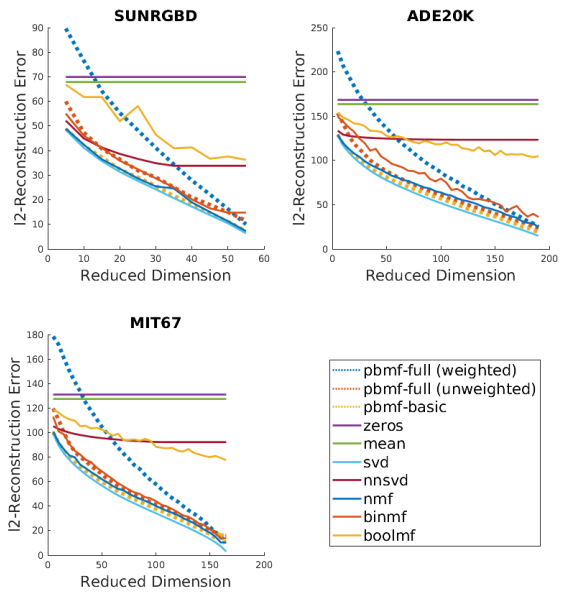


Figure 3: Reconstruction error between a recovered and groundtruth matrix as the dimensionality of the reduced representation is varied

4.3 Multi-Object Recognition from Scene Images

In addition to noise from the lossy PBMF, we also need to consider noise resulting from mapping visual data to objects. We consider the task of tagging images with their constituent objects. We use ScenarioNet to predict the scenario encoding matrix $\hat{H}$ and try to recover *an approximate hypothesis* about which objects are present in a scene by recovering the object-scene data matrix A using the learned scenario dictionary W and predicted encoding matrix $\hat{H}$: $A \approx W\hat{H}$. This recovery gives us a list of the *possible* objects in a scene. We compare against a finetuned object detector [Redmon and Farhadi, 2017] and VGG-16 finetuned for multi-object recognition. We use the macro (averaged) area under the precision-recall curve as our metric since the data is very imbalanced for rare objects. In Table 1, we show results for when we consider objects that appear in at least 1% of data (very rare) and 5% of data (rare) to show how imbalance affects both methods. The object detector works well when we have large, complete objects, but ADE20K tends to contain smaller objects and parts-of-objects, so the VGG-based nets outperform the object detector on this data, and perform worse on SUNRGBD. Likewise, there is greater labor cost associated with training the YOLO net because it requires bounding box information, which the other methods do not. VGG-Objects and ScenarioNet perform similarly despite PBMF being lossy and the output of ScenarioNet being 2-3 times smaller in dimensionality. Interestingly, ScenarioNet performs better than VGG-Objects on the SUNRGBD-5% task. We believe ScenarioNet performs well because it excels at capturing context and because it is easier to recognize scenarios (defined by a few key objects) than individual objects. ScenarioNet has several advantages over individual object-based methods: it finds relationships between objects and captures global scene information.

4.4 Scene Classification

We now consider the task of scene classification where we care more about global scene information than local objects. In the following sections, we compare ScenarioNet to other object-based representations, baseline CNNs, compressed CNNs, and other mid-level features. Results are reported in Table 2. For experiments not involving a CNN, we train a logistic regression model on top of the given features.

	SUNRGBD		ADE20K	
Method	1% (55 Objs)	5% (16 Objs)	1% (193 Objs)	5% (50 Objs)
Random	0.066	0.152	0.070	0.171
Object Detection (YOLOv2)	0.442	0.633	0.379	0.587
VGG-Objects	0.369	0.574	0.475	0.696
ScenarioNet	**0.356**	**0.585**	**0.452**	**0.683**

Table 1: Macro-AUPRC for multi-object recognition

Method	Dimens.	SUNRGBD	ADE20K	MIT
Object-Based Representations				
Object Bank + PCA	8000	0.296	0.511	0.39
Object Detection (YOLOv2)	55/193/166	0.399	0.639	0.517
VGG-Objects	55/193/166	0.483	0.726	0.6187
Baseline CNNs				
AlexNet	4096	0.469	0.786	0.687
GoogLeNet	2048	0.541	0.796	0.737
VGG-16	4096	0.531	0.809	0.792
ResNet-50	1024	0.509	0.777	0.687
Dimensionality-Reducing and Lower-Parameter CNNs				
VGG-Reduced	25/70/70	0.458	0.787	0.722
VGG-GAP	512	0.486	0.767	0.779
VGG-GMP	512	0.463	0.786	0.723
Attribute-Based Representations				
SUN-Attribute	102	0.429	0.705	0.655
Classemes	2659	0.309	0.581	0.448
Meta-Classes	15232	0.36	0.635	0.525
Learned Mid-Level Visual Representations				
Mid-Level Patches	14070	N/A	N/A	0.381*
Mid-Level Vis. Elem.	67000	N/A	N/A	0.64*
DPM	N/A	N/A	N/A	0.304*
RBoW	N/A	N/A	N/A	0.379*
BoP	3350	N/A	N/A	0.461*
Discriminative Parts	4926	N/A	N/A	0.514*
Proposed Model				
ScenarioNet	**25/70/70**	**0.520**	**0.794**	**0.725**

Table 2: Scene classification accuracy; * denotes reported results

Comparison to Other Object-Based Representations

We first consider object-based representations. These include the same models as in Sec. 4.3 (using the object probabilities as features) and also Object Bank features [Li *et al.*, 2010] compressed to 8000 dimensions using PCA. ScenarioNet is better than all other object-based representations for scene classification despite its lower dimensionality. This suggests that scenarios are better at capturing global scene information than individual object-based approaches. This is partly because ScenarioNet is trained to jointly recognize objects and scenes, a key difference to the other methods.

Comparison to Baseline CNNs

CNNs are currently a very popular method for scene classification. We finetune AlexNet [Krizhevsky *et al.*, 2012], GoogLeNet [Szegedy *et al.*, 2015], and VGG-16 [Simonyan and Zisserman, 2014] models that have been pre-trained on the Places dataset [Zhou *et al.*, 2017a] as well as a ResNet-50 CNN [He *et al.*, 2016] pre-trained on ImageNet [Deng *et al.*, 2009]. Since ScenarioNet extends VGG-16, we focus on how these two nets compare. ScenarioNet tends to slightly underperform VGG-16 by about 1-2%. The most significant drop in performance is on the MIT dataset, but ScenarioNet is forced to learn scenarios on a smaller, noisier, and imbalanced subset of the training data (see Sec. 4.1). ScenarioNet has several advantages over VGG-16; it has fewer pa-

rameters, has the ability to explain its decisions, and can produce scenario encodings which are useful for tasks beyond scene classification. In the next section, we see that ScenarioNet generally performs better than VGG-16 nets that compress the feature space to the same dimensionality as ScenarioNet.

Comparison to Dim-Reducing and Low-Param. CNNs

We modify VGG-16 so the output of the final feature layer is the same dimensionality as our scenario-based representation (by shrinking the FC layers). ScenarioNet matches or outperforms the compressed VGG-16 net in all cases. This might be because ScenarioNet constrains the intermediate representation to have high-level meaning while the compressed-VGG net lacks such guidance, making it susceptible to finding worse local minimum. We also compare ScenarioNet to VGG nets which replace the double fully-connected layers with global average pooling (GAP) and global max pooling (GMP) layers. These nets contain roughly the same number of parameters as ScenarioNet. In five of six cases, we outperform or match the low-parameter nets.

Comparison to Mid-Level Representations

Finally, we compare against three other types of mid-level representations: attributes, mid-level visual patches, and parts-based models. Attributes are high-level semantic properties shared between multiple classes [Farhadi *et al.*, 2009]. We consider three attribute-like representations: SUN Attributes [Patterson and Hays, 2012], Classemes [Torresani *et al.*, 2010], and Meta-Classes [Bergamo and Torresani, 2012]. Several representations consider visually-distinct, meaningful mid-level patches [Singh *et al.*, 2012] and mid-level visual elements [Doersch *et al.*, 2013]. Finally, we consider parts-based models including the deformable parts model (DPM) [Pandey and Lazebnik, 2011], reconfigurable bags-of-words (RBoW) [Parizi *et al.*, 2012], bags-of-parts (BoP) [Juneja *et al.*, 2013], and discriminative parts [Sun and Ponce, 2013]. We outperform all of these methods, but it should be noted that for the non-attribute-based features, we use the reported results on the MIT dataset because the code to generate these features is either unavailable or prohibitively expensive to run on our machines. It should also be noted that these methods pre-date CNNs, and not all of the reported results include the use of training data augmentation while ScenarioNet does.

4.5 Content-Based Querying and Comparison

ScenarioNet is useful for content-based scene image retrieval because it can retrieve images satisfying a set of high-level criteria based on the scene category, scenarios, and objects present in an image (e.g., find images of scene category A OR B THAT CONTAIN scenarios X AND Y but EXCLUDE object Z). Often, we want to query for broad concepts and not individual objects. Scenarios offer a nice compromise between global (scene category) and local (object) information. It is easy for humans to examine the scenario dictionary and form complex queries because scenarios are low-dimensional and interpretable. Scenarios can also act as an efficient hashing mechanism because they are low-dimensional and approximately binary, so memory requirements are low and retrieval can be performed in an efficient manner.

In Table 3, we evaluate ScenarioNet for complex content-based scene image retrieval. We form 500 random queries, each consisting of a desired scene class, two objects that should be present, and one object that should be absent but frequently co-occurs with the other two objects, i.e. $(SC \cap O_1 \cap O_2 \cap \neg O_3)$. We do not consider querying against scenarios for this task because no other method is capable of recognizing scenarios. We measure the relevance of a returned image as the proportion of query terms that are satisifed. We compute the normalized discounted cumulative gain for the top-5 result images for each query. ScenarioNet is very competive with the other methods, matching VGG-Objects for the best performance on ADE20K, and coming very close to both baselines on SUNRGBD.

Method	SUNRGBD	ADE20K
Random	0.302	0.313
Object Detection (YOLOv2)	0.679	0.760
VGG-Objects	0.686	0.799
ScenarioNet	**0.652**	**0.799**

Table 3: NDCG@5 for retrieving images of a given class containing 2 specific objects and not containing a third highly-correlated object

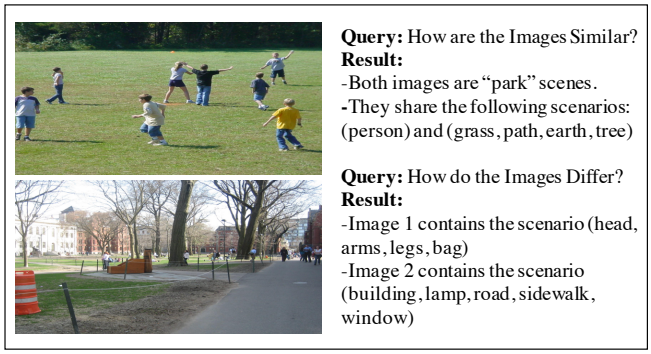


Figure 4: Using ScenarioNet to find high-level similarities and differences between two images.

ScenarioNet is also useful for generating a quick overview of the similarities and differences between two scene images without relying on (often unnecessary) information about individual objects. Fig. 4 shows an example.

5 Conclusions

We introduced scenarios as a new way of representing scenes. The scenario is a simple data-driven representation based on sets of frequently co-occurring objects. We provided a method for learning scenarios from data by combining PBMF with CNNs to form the ScenarioNet. Our experiments showed that a single ScenarioNet model can perform scene classification, scenario recognition, multi-object recognition, content-based scene image retrieval, and content-based image comparison with performance comparable to or better than existing models. We showed that scenarios have several advantages over individual object-based representations; specifically, they are lower-dimensional, capture global scene context, and find relationships between objects. We also discussed and demonstrated the computational efficiency and interpretability of ScenarioNet compared to traditional CNNs. We believe ScenarioNet provides a strong first step towards constructing explainable and trustworthy models for safety-critical applications related to scene understanding (e.g., robotics, human-machine teaming, surveillance, and self-driving cars). However, much work remains, including evaluating the utility of scenarios in human studies and figuring out how ScenarioNet can be used in dynamic and interactive settings.

Acknowledgments

This work is partly supported by the Air Force Office of Scientific Research (AFOSR) under the Dynamic Data-Driven Application Systems (DDDAS) program. This material is based upon work supported by the National Science Foundation Graduate Research Fellowship under Grant No. DGE-1433187.

References

[Bergamo and Torresani, 2012] Alessandro Bergamo and Lorenzo Torresani. Meta-class features for large-scale object categorization on a budget. In *CVPR*, 2012.

[Choi *et al.*, 2012] Myung Jin Choi, Antonio Torralba, and Alan S Willsky. A tree-based context model for object recognition. *IEEE TPAMI*, 2012.

[Cinbis and Sclaroff, 2012] Ramazan Cinbis and Stan Sclaroff. Contextual object detection using set-based classification. *ECCV*, 2012.

[Deng *et al.*, 2009] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*. IEEE, 2009.

[Ding *et al.*, 2006] Chris Ding, Tao Li, Wei Peng, and Haesun Park. Orthogonal nonnegative matrix t-factorizations for clustering. In *SIGKDD*, 2006.

[Doersch *et al.*, 2013] Carl Doersch, Abhinav Gupta, and Alexei A Efros. Mid-level visual element discovery as discriminative mode seeking. In *NIPS*, 2013.

[Fan *et al.*, 2008] Jianping Fan, Yuli Gao, and Hangzai Luo. Integrating concept ontology and multitask learning to achieve more effective classifier training for multilevel image annotation. *IEEE TIP*, 2008.

[Farhadi *et al.*, 2009] Ali Farhadi, Ian Endres, Derek Hoiem, and David Forsyth. Describing objects by their attributes. In *CVPR*, 2009.

[Feng and Bhanu, 2016] Linan Feng and Bir Bhanu. Semantic concept co-occurrence patterns for image annotation and retrieval. *IEEE TPAMI*, 2016.

[He *et al.*, 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[Hendricks *et al.*, 2016] Lisa Anne Hendricks, Zeynep Akata, Marcus Rohrbach, Jeff Donahue, Bernt Schiele, and Trevor Darrell. Generating visual explanations. In *ECCV*, 2016.

[Johnson *et al.*, 2016] Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. *arXiv*, 2016.

[Juneja *et al.*, 2013] Mayank Juneja, Andrea Vedaldi, CV Jawahar, and Andrew Zisserman. Blocks that shout: Distinctive parts for scene classification. In *CVPR*, 2013.

[Krizhevsky *et al.*, 2012] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[Lan *et al.*, 2013] Tian Lan, Michalis Raptis, Leonid Sigal, and Greg Mori. From subcategories to visual composites. In *ICCV*, 2013.

[Lengerich *et al.*, 2017] Benjamin J Lengerich, Sandeep Konam, Eric P Xing, Stephanie Rosenthal, and Manuela Veloso. Visual explanations for convolutional neural networks via input resampling. *ICML Workshop on Visualization in Deep Learning*, 2017.

[Li *et al.*, 2010] Li-Jia Li, Hao Su, Li Fei-Fei, and Eric P Xing. Object bank: A high-level image representation for scene classification. In *NIPS*, 2010.

[Li *et al.*, 2012] Congcong Li, Devi Parikh, and Tsuhan Chen. Automatic discovery of groups of objects for scene understanding. In *CVPR*, 2012.

[Miettinen *et al.*, 2008] Pauli Miettinen, Taneli Mielikäinen, Aristides Gionis, Gautam Das, and Heikki Mannila. The discrete basis problem. *IEEE TKDE*, 2008.

[Oquab *et al.*, 2015] Maxime Oquab, Léon Bottou, Ivan Laptev, and Josef Sivic. Is object localization for free? In *CVPR*, 2015.

[Paatero and Tapper, 1994] Pentti Paatero and Unto Tapper. Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics*, 1994.

[Pandey and Lazebnik, 2011] Megha Pandey and Svetlana Lazebnik. Scene recognition and weakly supervised object localization with deformable part-based models. In *ICCV*, 2011.

[Parizi *et al.*, 2012] Sobhan Naderi Parizi, John G Oberlin, and Pedro F Felzenszwalb. Reconfigurable models for scene recognition. In *CVPR*, 2012.

[Park *et al.*, 2016] Dong Huk Park, Lisa Anne Hendricks, Zeynep Akata, Bernt Schiele, Trevor Darrell, and Marcus Rohrbach. Attentive explanations. *arXiv*, 2016.

[Patterson and Hays, 2012] Genevieve Patterson and James Hays. Sun attribute database: Discovering, annotating, and recognizing scene attributes. In *CVPR*, 2012.

[Quattoni and Torralba, 2009] Ariadna Quattoni and Antonio Torralba. Recognizing indoor scenes. In *CVPR*, 2009.

[Rabinovich *et al.*, 2007] Andrew Rabinovich, Andrea Vedaldi, Carolina Galleguillos, Eric Wiewiora, and Serge Belongie. Objects in context. In *ICCV*, 2007.

[Redmon and Farhadi, 2017] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. *CVPR*, 2017.

[Ribeiro *et al.*, 2016] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Why should i trust you? In *SIGKDD*, 2016.

[Selvaraju *et al.*, 2017] Ramprasaath Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. *ICCV*, 2017.

[Simonyan and Zisserman, 2014] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv*, 2014.

[Singh *et al.*, 2012] Saurabh Singh, Abhinav Gupta, and Alexei A Efros. Unsupervised discovery of mid-level discriminative patches. In *ECCV*. 2012.

[Song *et al.*, 2015] Shuran Song, Samuel P Lichtenberg, and Jianxiong Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *CVPR*, 2015.

[Sun and Ponce, 2013] Jian Sun and Jean Ponce. Learning discriminative part detectors for image classification and cosegmentation. In *ICCV*, 2013.

[Szegedy *et al.*, 2015] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.

[Torresani *et al.*, 2010] Lorenzo Torresani, Martin Szummer, and Andrew Fitzgibbon. Efficient object category recognition using classemes. *ECCV*, 2010.

[Zhang *et al.*, 2007] Zhongyuan Zhang, Tao Li, Chris Ding, and Xiangsun Zhang. Binary matrix factorization with applications. In *ICDM*, 2007.

[Zhou *et al.*, 2016] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *CVPR*, 2016.

[Zhou *et al.*, 2017a] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE TPAMI*, 2017.

[Zhou *et al.*, 2017b] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *CVPR*, 2017.

39

Explaining Deep Adaptive Programs via Reward Decomposition

Martin Erwig[1], **Alan Fern**[1], **Magesh Murali**[1], **Anurag Koul**[1]
[1]School of EECS
Oregon State University

Abstract

Adaptation Based Programming (ABP) allows programmers to employ "choice points" at program locations where they are uncertain about how to best code the program logic. Reinforcement learning (RL) is then used to automatically learn to make choice-point decisions to optimize the reward achieved by the program. In this paper, we describe an approach to explaining the learned decisions of adaptive programs. The key idea is to decompose rewards into rewards of different types and to explain choices of actions by a minimal subset of rewards that are sufficient for distinguishing the benefit of one action over another. We demonstrate the explanation approach through a small case study, which shows the general promise of the approach and highlights future research questions.

1 Introduction

Programmers are often faced with uncertain choices about program logic that can significantly impact the quality of the resulting program. Adaptation Based Programming (ABP) [Bauer *et al.*, 2011] is a programming paradigm in which programmers can explicitly declare their uncertainty about such choices via the insertion of adaptive variables into the program. We refer to programs with such adaptive variables as adaptive programs. In addition, adaptive programs include reward statements that indicate the quality of program executions. The key idea of ABP is to use reinforcement learning (RL) to optimize the decisions of adaptive variables such that the reward acquired during program executions is optimized. In particular, in this work we focus on deep adaptive programs (DAP), where choice point decisions are automatically learned via deep neural networks.

To gain confidence in learned decisions of DAPs, it is important to produce explanations for those decisions. In this paper, we focus on explaining the reasons that one alternative A of an adaptive variable was preferred to another alternative B. The key questions are: (1) What form should the explanations take?, and (2) How to adjust the learning algorithms so that such explanations can be produced?

To address the first question, we allow the programmer to provide a simple form of program annotation. The annotations specify reward types, which label each reward asserted by the program by one of a fixed set of types. For example, in a real-time strategy game, one type of reward would be related to damage inflicted on the enemy and another type would be based on damage the enemy inflicts on us. These types might be further decomposed into damage inflicted on or taken from specific types of game units. The goal of learning is still to maximize the total reward, accumulated across all types. However, the types can provide significant insight into decisions. Indeed without type information, an explanation for why A was selected over B might be the trivial explanation that it is predicted that selecting A will lead to more future reward than B. Rather, with type information, we open the possibility to comparing A and B in terms of the predicted future reward of each type, which provides a finer-grained view of the tradeoffs going into the decisions and significantly more insight.

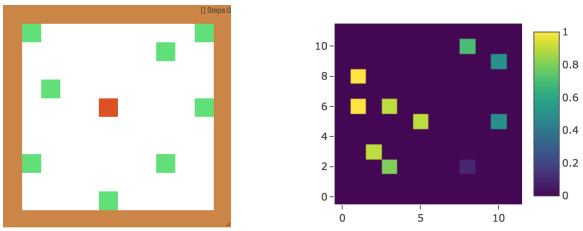
To instantiate this idea we define the notion of reward difference explanations (RDXs), which explain why A is preferred to B. An RDX is simply a vector that records the difference in predicted future reward for each reward type when taking A versus B and then following the program thereafter. We further define the notion of a minimal sufficient explanation for an RDX as the minimal number of reward components needed to prove that A is preferred to B.

To address the second question, we present a SARSA-style algorithm for learning policies to control the decisions of adaptive variables and at the same time learning information needed to produce RDXs for any pair of decisions. This idea has been fully implemented within a Python-based ABP library for deep adaptive programs. We demonstrate this algorithm and explanation approach via a case study on a synthetic problem designed to include decisions that are not obvious to a human and hence require explanation. The results show that the approach is able to provide useful explanations and that the explanations tend to be quite compact.

2 Adaptation-Based Programming

To illustrate explanations in DAPs let us consider the following example problem shown in Figure 1. The agent in this example moves across a grid with the goal of collecting as many fruits as possible within 100 steps. The locations of the fruits are fixed. The environment also contains lightning events that occur randomly with every step with a probability

that can depend on the grid location. If the agent is struck by lightening, it dies, and the game ends.



(a) (b)

Figure 1: (a) Grid world for the fruit collection problem; red indicates the position of the agent, and green indicates the position of the fruits. (b) A heat map that shows the probabilities of locations of lightening events.

The fruit collection problem can be formulated as an ABP program as shown in Listing 1, in which the problem is represented as an OpenAI gym environment `env`. In addition to state initialization, this object provides the method `step` for making state transitions. The object `move` represents an adaptive variable that offers four choices for moving in different directions, and it contains methods for making RL-informed choices and collecting rewards.

```
state = env.reset()
move = Adaptive(choices = [UP, DOWN, LEFT, RIGHT])

while not done:
  direction = move.choose(state)
  state, reward, done = env.step(direction)
  move.adapt(reward)
```

Listing 1: Adaptive program for the fruit collection problem

The `step` function performs an action provided as input and returns the resulting state, the reward received, and a boolean indicating whether the current episode has ended. In this example, this boolean is `True` when the agent has collected all the fruits, the maximum number steps of 100 has been reached, or the agent is struck by lightening. The `state` object is a two-dimensional boolean matrix representing the position of the agent and the location of the fruits in the environment.

In each of the 100 steps available to the agent, the method `choose` first selects a direction to move, then the method `step` moves the agent in that direction. After that, the reward resulting from this step is accumulated with the method `adapt`. The adaptive variable uses RL to choose the correct value for the adaptive based on the current state [Bauer *et al.*, 2011]. We use a modified version of the SARSA RL algorithm in our approach. Using standard RL terminology, the adaptive variables are treated as decision points, and the values of adaptive variables are treated as actions that can be chosen at those points. Each time a decision point is encountered the "state" of the decision point is used as a basis for the action/value selection. The library allows the user to define arbitrary features of decision points that can be used as the state representation.

We note that while this example program contains only a single adaptive variable, for illustration purposes, our library supports programs that include an arbitrary number of adaptive variables. Using multiple adaptive variables allows for modeling a wide range of agent decision-making architectures such as those used in hierarchical RL [Dietterich, 2000; Parr and Russell, 1998]. The ABP framework also generalizes prior work on "partial programming" [Andre and Russell, 2002] where the semantics were defined relative to an external MDP rather than just the adaptive program.

After the initial training period, the adaptive variable is associated with a Q-function. The Q-function $Q(s, a)$ maps the state s of a decision point and choice a to a numeric value. In particular, $Q(s, a)$ is an estimate of the expected future reward that will be achieved if choice a is selected and then the adaptive program is followed thereafter. Given a decision point, an associated Q-function, the adaptive program will select the choice that maximizes the learned Q-function. In our current library, the Q-function for each adaptive variable is represented via a neural network that takes the state as input and outputs the Q-value for each possible choice. In our fruit collection example, the neural network input is the raw map, where a one-hot encoding is used to represent the contents of each cell.

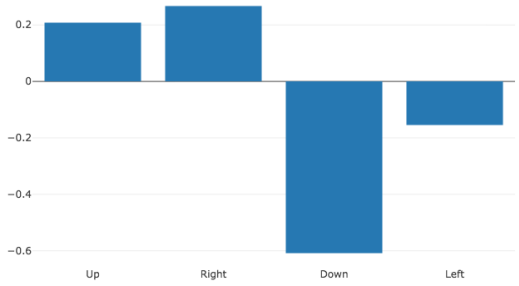


Figure 2: Q-values for the actions in the state shown in Figure 1(a).

Figure 2 illustrates a situation in which the agent has to move right in order to maximize the reward. However, having a single Q-value does not give us a sufficient explanation. It compresses the knowledge about the problem into a single value. It is not clear why the agent made the decision: was the agent "more worried" about getting struck by lightning when moving in the other directions? Which fruit are they trying to collect?

3 Explaining via Reward Decomposition

Our main strategy for obtaining better explanations is to notice that in many RL environments the reward can be broken down into multiple reward types, each corresponding to semantically distinct ways of acquiring reward. This allows for decomposing the overall reward function into a number of different reward component functions, which sum up to the total reward. Often natural reward decompositions are obvious to a programmer or environment designer, since defining the reward function requires thinking about the distinct circumstances under which reward is generated. The traditional RL paradigm, however, does not allow a developer to explicitly expose that knowledge.

In the fruit collection example we can decompose the reward into 9 different reward types, one for each location of the fruit plus one for when the traveller gets hit by lightning. In order to support this, we have extended the ABP library to use reward annotations. The program for the fruit collection problem can be extended by annotations that use locations to distinguish the different fruits and a tag for the lightening reward.

Given a reward decomposition, we can adjust the learning algorithm so that it learns a distinct Q-function for each reward component, each predicting the future reward of the corresponding component only, see Listing 2. It is straightforward to see that the sum of the individual component Q-functions is equal to the Q-function for the overall cumulative reward. In our fruit collection problem, there will be a Q-function for each fruit location and for lightning. As an example, the component Q-function for the lightning reward type gives the expected future reward (negative) due to lightning strikes after taking an a particular action in a particular state and then following the program thereafter.

```
rewards = []
lightning_pos = generate_lightning()

if agent_location in fruit_locations:
   rewards.append((agent_location, 2))

if agent_location in lightning_pos:
  rewards.append(("Lightning Strike", -1))
```

Listing 2: Decomposed reward generation in each step

The part of the adaptive program that represents the core of the example needs only a minor modification. The only change is the replacement of the single call to `adapt`, which uses one reward value, by a loop that iterates over a list of rewards and corresponding reward types, see Listing 3. In this scenario, this list always contains either zero or one element.

```
state = env.reset()
move = Adaptive(choices = [UP, DOWN, LEFT, RIGHT])

while not done:
  direction = move.choose(state)
  state, rewards, done = env.step(direction)
  for typedReward in rewards:
     move.adapt(typedReward)
```

Listing 3: Adaptive Program using reward decomposition

Given a set of component Q-functions for each reward type, decisions are made by selecting the action that maximizes the cumulative Q-function, which is equal to the sum of the components. Thus, learning should both ensure that the cumulative Q-function converges to the optimal Q-function for the overall reward, while also ensuring that the component Q-functions converge to their correct values. A similar learning problem has been studied in prior work [Russell and Zimdars, 2003] on multi-agent learning. In that work, the total reward was decomposed across multiple learning agents that were controlled by a centralized policy. If we equate the agents with reward types, the two formulations become identical. Thus, we use the SARSA-style algorithm develop in that work in the context of ABP. The algorithm uses on-policy SARSA learning [Sutton and Barto, 1998] to update each of the component Q-functions and drives exploration using an ϵ-greedy policy based on the cumulative Q-function.

We now consider an example decision of the agent for the state shown in Figure 1(a). This decision can be explained much better using the decomposed Q-values and reward types. If we take a look at the decomposed Q-values and reward types in Figure 3, we can observe that moving right is better than moving down because moving right has more positive rewards for almost all fruits and at the same time less of a chance of being struck by lightening than moving down. However, the decomposition of the rewards itself does only part of the job. For example, it is not so clear why moving right is better than moving up.

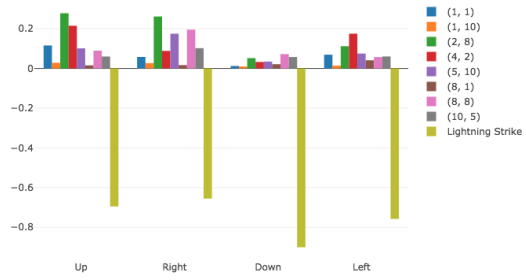


Figure 3: The Q-values from Figure 2 decomposed by different reward types for fruit locations and lightening.

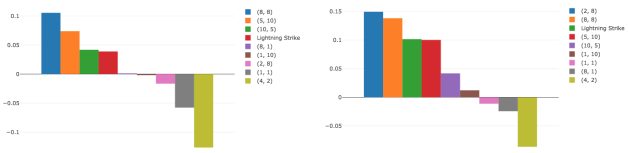
To better compare the decomposed Q-values for different actions, we can compute their difference and then order the result by magnitude. In the context of decomposed Q-values, $Q_\pi(s, A)$ is a vector that returns the component Q-values for each reward type for the current policy π (generally the ϵ-greedy policy) in state s and action A. We write $Q^R_\pi(s, A)$ for the Q-value component of reward type R. We call the reward-type-indexed difference between the decomposed reward for two actions A and B their *Reward Difference Explanation* (RDX).

$$\Delta_\pi(s, A, B) = Q_\pi(s, A) - Q_\pi(s, B)$$

A positive value in an RDX mapping for a reward type R indicates that the adaptive variable predicts a higher reward of type R for A than for B. Correspondingly, a negative value indicates that action B promises higher reward of type R.

To compare moving `RIGHT` with moving `UP` for the state shown in Figure 1(a) we compute the RDX $\Delta_\pi(s, \texttt{RIGHT}, \texttt{UP})$ and visualize the resulting mapping in Figure 4(a), ordering the RDX components by magnitude. To convince ourselves that `RIGHT` is indeed the better action, we have to ensure that the sum of the positive reward differences, representing the advantages of `RIGHT` over `UP` is greater than the sum of the negative ones, which represent the disadvantages.

The four reward types with negative values sum up to -0.20, while the sum of the first three positive reward yields 0.22, which is enough to be sure that `RIGHT` is expected to be the preferred action. In other words, moving `RIGHT` promises enough reward for collecting the three fruits at the shown locations that makes up for any other expected disadvantages. This example also shows that we don't have to consider all positive reward types to make this determination. This phenomenon is even more salient when we compare the actions `RIGHT` and `LEFT`. In this case, the reward for one fruit with a



(a) $\Delta_\pi(s, \texttt{RIGHT}, \texttt{UP})$ (b) $\Delta_\pi(s, \texttt{RIGHT}, \texttt{LEFT})$

Figure 4: Reward Difference Explanations for two action pairs.

value of 0.14 is sufficient to cover all expected losses, which are about -0.12.

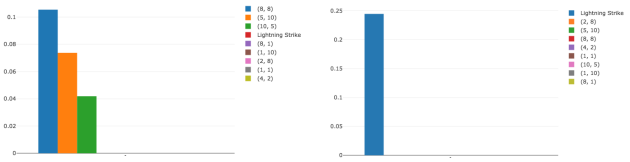
This observation leads to the definition of a *minimal sufficient explanation* (MSX), which is the minimal subset of $\Delta_\pi(s, A, B)$ whose sum of rewards exceeds the sum of all negative rewards from $\Delta_\pi(s, A, B)$. We denote the sum of all individual rewards of a reward decomposition mapping Q with $\Sigma(Q) = \sum_{(r,x)\in Q} x$. Now with:

$$R^-(Q) = \Sigma(\{(r, x) \in Q \wedge x < 0\})$$
$$Sub^+(Q) = \{S \subseteq Q \mid (r, x) \in S \Rightarrow x > 0\}$$

we can define:

$$\mu_\pi(s, A, B) = S \in Sub^+(Q) : \Sigma(S) > R^-(Q) \wedge |S| \text{ is minimal}$$
$$\text{where } Q = \Delta_\pi(s, A, B)$$

The minimal sufficient explanations for the reward differences from Figure 4 are shown in Figure 5. They allow the user of an explanation to focus on a subset of reward types.



(a) $\mu_\pi(s, \texttt{RIGHT}, \texttt{UP})$ (b) $\mu_\pi(s, \texttt{RIGHT}, \texttt{LEFT})$

Figure 5: Minimal Sufficient Explanations for two action pairs.

The concept of minimal sufficient explanations (called MSE) was also introduced in closely related work on explanations for optimal Markov Decision Process (MDP) policies [Khan *et al.*, 2009]. There are several key distinctions between that work and ours. First, that work assumed a perfect model of the environment and approached the problem from an automated planning perspective. Rather, we are in an RL setting, which introduces challenges due to the lack of a model. Second, that work focused on explaining the optimal decision of an optimal policy. In particular, the notion of MSE used there required that the MSE proved that the optimal action was better than all other actions. Such explanations will often be significantly larger than our pairwise explanations, since an optimal action may be better than other actions for orthogonal reasons. In such cases, their MSE must include all of those reasons. Finally, the prior work did not introduce the notion of reward decomposition. Rather they relied on a generic, but extremely fine-grained decomposition that effectively defined a reward component for each state. Such an approach is quite limited to domains with small state spaces.

One question that remains is: How effective are minimal sufficient explanations? In the fruit collection scenario we have 9 different reward types, and so it might be possible for any $\mu_\pi(s, A, B)$ to vary in size from 0 to 8. (Here a size of 0 means that the RDX has no negative component and that action A is better in every aspect.) To answer this question we computed the frequencies of the sizes $|\mu_\pi(s, A, B)|$ for all action pairs for an optimal policy π. As Figure 6 shows, in the vast majority of cases a single reward type is sufficient to explain an action, which shows that the technique is quite effective in focusing the attention of users on particular parts of reward components.

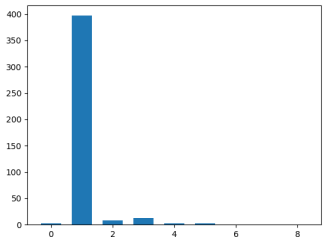


Figure 6: Frequencies of $|\mu_\pi(s, A, B)|$

4 Conclusions

We have demonstrated an approach to explain machine-learned decisions in the context of adaptation-based programming by decomposing rewards into labeled parts. Decomposition and reward typing together provide the opportunity to justify decisions by comparing advantages and disadvantages of decisions with respect to different reward types. This idea is formalized through reward difference explanations. In addition, by employing the concept of minimal sufficient explanations we can often generate explanations that need to mention only few of the potentially large set of different reward types. As our analysis has shown, in the example scenario the vast majority of explanations require mention of only one reward type.

Acknowledgements

This work is partially supported by the National Science Foundation under the grant CCF-1717300 and by DARPA under the grant N66001-17-2-4030.

References

[Andre and Russell, 2002] David Andre and Stuart J Russell. State abstraction for programmable reinforcement learning agents. In *AAAI/IAAI*, pages 119–125, 2002.

[Bauer *et al.*, 2011] Tim Bauer, Martin Erwig, Alan Fern, and Jervis Pinto. Adaptation-based programming in java. In *Proceedings of the 20th ACM SIGPLAN workshop on Partial evaluation and program manipulation*, pages 81–90. ACM, 2011.

[Dietterich, 2000] Thomas G Dietterich. Hierarchical reinforcement learning with the maxq value function decomposition. *Journal of Artificial Intelligence Research*, 13:227–303, 2000.

[Khan *et al.*, 2009] Omar Zia Khan, Pascal Poupart, and James P Black. Minimal sufficient explanations for factored markov decision processes. In *ICAPS*, 2009.

[Parr and Russell, 1998] Ronald Parr and Stuart J Russell. Reinforcement learning with hierarchies of machines. In *Advances in neural information processing systems*, pages 1043–1049, 1998.

[Russell and Zimdars, 2003] Stuart J Russell and Andrew Zimdars. Q-decomposition for reinforcement learning agents. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 656–663, 2003.

[Sutton and Barto, 1998] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

Generating Natural Language Explanations for Visual Question Answering Using Scene Graphs and Visual Attention

Shalini Ghosh[*], **Giedrius Burachas**[†], **Arijit Ray**[†], **Avi Ziskind**[†]
[*]firstname.lastname@samsung.com, [†]firstname.lastname@sri.com
[*]Samsung Research, [†]SRI International

Abstract

In this paper, we present a novel approach for the task of eXplainable Question Answering (XQA), i.e., generating natural language (NL) explanations for the Visual Question Answering (VQA) problem. We generate NL explanations comprising of the evidence to support the answer to a question asked to an image using two sources of information: (a) annotations of entities in an image (e.g., object labels, region descriptions, relation phrases) generated from the scene graph of the image, and (b) the attention map generated by a VQA model when answering the question. We show how combining the visual attention map with the NL representation of relevant scene graph entities, carefully selected using a language model, can give reasonable textual explanations without the need of any additional collected data (explanation captions, etc). We run our algorithms on the Visual Genome (VG) dataset and conduct internal user-studies to demonstrate the efficacy of our approach over a strong baseline. We have also released a live web demo showcasing our VQA and textual explanation generation using scene graphs and visual attention.[1]

1 Introduction

Visual Question Answering (VQA) [Antol *et al.*, 2015], the task of answering natural language questions on images, has garnered a lot of interest as an AI-complete task. While impressive strides have been made on this task using deep networks [Kazemi and Elqursh, 2017; Teney *et al.*, 2017], they are notorious for being opaque/black-boxed to a non-expert user, thus making it hard to understand when/why it predicts an incorrect answer. There have been attempts to make VQA systems more human-like [Ray *et al.*, 2016] and on how they can hold conversations if one desires further questioning [Ray, 2017] [Das *et al.*, 2016]. However, VQA/conversational agents still cannot explain in natural language why they made a certain decision. This raises issues of trust and reliability since the user cannot judge when to trust the predictions of the model or not. In this paper, we focus on a natural language solution to the eXplainable Question Answering (XQA) task, the task of explaining the answer that was provided — specifically, our goal is to automatically generate a natural language sentence that provides evidence to support the answer predicted by a VQA model. Ideally, such an explanation will help people judge and/or trust the answer provided better.

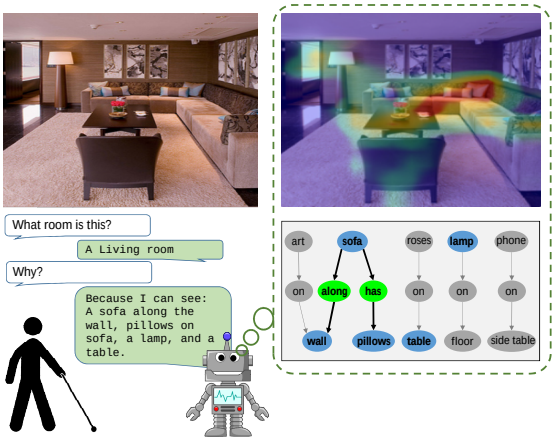


Figure 1: Example Natural Language explanation for an answer to a visually-grounded question. Our approach uses the attention map generated by a visual question-answering model (top right) to identify the relevant components of a scene graph (bottom right) that can be used to provide a justification for the answer given by the model.

Figure 1 illustrates a summary of our objective. Asked a question, "What room is this?", to which the answer is "living room", the visually impaired user requests for an explanation in order to ensure the answer is correct. Our Explainable VQA Agent uses the attention heatmap (regions of interest as suggested by the model while answering the question) to pick out relevant information from an annotated image scene graph to generate a natural language explanation phrase — "Because I can see: sofa along the wall, pillows on sofa, a lamp, and a table." Our proposed algorithm uses the visual attention map as a guide to identify the relevant entities from the scene graph (where entities could be objects, relations or descriptions), and then uses NLP techniques using language models to compose

[*]Work done while the first author was at SRI International.
[1]https://xai.nautilus.optiputer.net/

a NL representation of those entities to generate a NL explanation for the VQA answer. By conducting a small-scale user study, we show evidence that such an approach can generate reasonable textual explanations for answers to questions on images in the Visual Genome Dataset.

Section 2 covers the relevant background and some related work. Section 3 outlines the core algorithms, while Section 4 gives some example explanations generated by our algorithm. We performed some initial experiments using the Visual Genome (VG) dataset to demonstrate the effectiveness of our method — an analysis of those results are described in Section 5. Section 6 discusses related approaches, and finally Section 7 concludes the paper and gives an overview of possible future work in this area.

Figure 2: Salient parts of an Visual Genome image highlighted by the attention layers of our VQA Model, corresponding to the question/answer pair "What is this game? Tennis".

2 Background

In this section, we briefly describe some of the key models and concepts used in this paper.

2.1 Visual Question Answering and Attention Layers

Visual Question Answering is the task of answering natural language questions about an image. This requires simultaneous understanding of textual and visual semantics — deep neural networks have made impressive strides at this task. We use the VQA architecture outlined in Figure 3. Our model takes as input a 224 x 224 RGB image and a question of at most 15 words. The image is encoded using a ResNet152 [He *et al.*, 2015] to get a 7x7x2048 image feature representation. The question is encoded using an LSTM which takes in the GloVe [Pennington *et al.*, 2014] word embeddings of the words, one word at a time. The final LSTM state is used to represent the question features. The attention layer takes in the question and image feature representations and outputs a set of weights to attend on the image features. The weighted image features, concatenated with the question representation, is used to predict the final answer from a set of 3000 answer choices.

2.2 Scene Graph

The scene graph for an image is the graphical representation of its contents, where the nodes are the depicted objects and the edges are the relationships between them (e.g. as shown in Figure 1). Some scene graphs also contain region descriptions (more detailed annotations of an object, or a description of a region containing multiple interacting objects). For example, the Visual Genome data has region descriptions in addition to labeled objects/relations, a sample of which is shown in Figure 4. However, most methods for generating scene graphs [Lu *et al.*, 2016] generate graphs with only objects and relations, but without region descriptions. Depending on the type of scene graph available, we designed two variants of the explanation candidate generation models: (1) one that generates NL explanations based on region descriptions, and (2) one that generates NL explanations based on objects and relations.

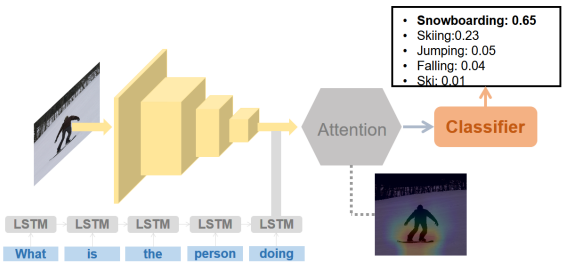


Figure 3: We use a simplistic VQA architecture which is very similar to the SA3 model by Google [Kazemi and Elqursh, 2017]. The model takes as input a 224 x 224 image and a natural language question (words encoded as 300 dimensional GloVe [Pennington *et al.*, 2014] vectors) and outputs a sigmoidal probability distribution over 3000 answer choices. The attention layer weighs a 7 x 7 x 2048 convolutional map, which is weighted averaged to get a 7 x 7 heatmap. The 7 x 7 heatmap is resized to 224 x 224 to get the attention heatmap for the image.

2.3 Web Language Model

Given a sequence of words, language models estimate the probability of observing another sequence of words following it. There are various language models available, typically trained on large scale corpora (e.g., Web news data, Wikipedia), which give robust estimates of conditional probabilities of observing one text segment in the context of another text segment [Józefowicz *et al.*, 2016]. We use the Web language model service AzureLM[2] to get robust estimates of conditional probabilities of text segments.

3 Algorithm

One of the key insights in our approach is the observation that the scene graph of an image can be very useful in generating explanations, something that has not been explored in previous work on generation visual explanations. We use the scene graph to retrieve the set of relevant entities for an image, where an entity refers to either an object, relation or region description, and then consider the natural language phrases corresponding to those entities. Note that each entity in the scene graph has an associated bounding box. For example, for the tennis image in Figure 4, the bounding box shown in the figure is associated with the region description "a tennis player hitting a ball".

[2]https://azure.microsoft.com/en-us/services/cognitive-services/web-language-model/

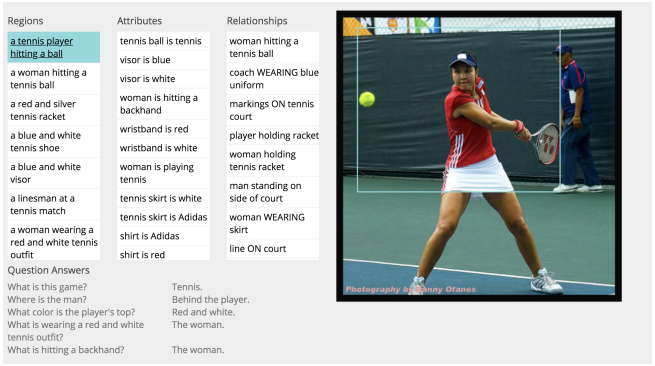


Figure 4: Scene graph of an image from Visual Genome data, showing object attributes, relation phrases and descriptions of regions [Krishna *et al.*, 2016].

For a given image and a question/answer pair, we use the heatmap generated by the visual attention layer to identify the parts of the image that are relevant and retrieve the bounding boxes with a high degree of overlap with these regions. For example, for the attention map shown in Figure 2, corresponding to the question/answer pair "What is this game? Tennis", the region of the image containing the tennis racket is highlighted, thus identifying this object within the scene graph as relevant for the explanation. We rank the most important entities from the scene graph using a score function, which includes, among other things: (1) The degree of overlap of the bounding box with the active region of the attention map, and (2) An estimate of the relevance of the NL representation of the entity w.r.t. the question/answer.

Our composite score helps us identify entities with NL representations that are deemed relevant to the explanation of the question/answer pair, both from the point of view of the visual attention model as well as the language model. E.g., for Figure 4, a region description with high relevance score (from both the visual model and language model) is "a tennis player hitting a ball" — we use this high-scoring region description to generate the final explanation "The picture shows: a tennis player hitting a ball".

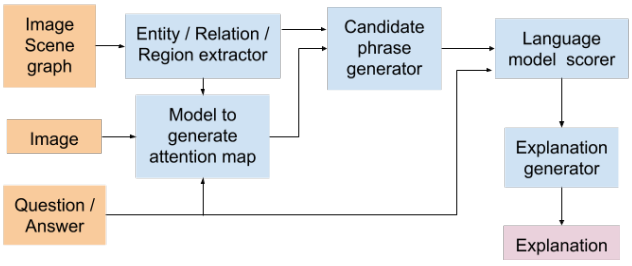


Figure 5: Proposed Explanation Generation Pipeline.

Figure 5 shows the overall flow of explanation generation used in our approach, where we generate explanations based on inferences by a language model and the visual attention layer. When we have region descriptions corresponding to an image, along with the visual attention model, we use the approach in Algorithm 1 to generate the explanations. The score function used to rank regions is:

$$\begin{aligned} score(D, QA) &= attentionScore(R(D)|QA) \\ &\times \ lmScore(D|QA) \\ &\times \ sqrt(len(D)) \\ &\times \ 1/log(area(R(D))) \end{aligned} \quad (1)$$

where D corresponds to a region description, $R(D)$ is the bounding box of that region, $attention(R(D)|QA)$ is the total attention score from the visual attention layer for region R(D) for question/answer Q/A, $lmScore(D|QA)$ is the language model score of text of D in the context of the text of QA, $len(D)$ is the length of description D and $area(R(D))$ is the area of R(D). Given two regions for related areas, this score function selects the region with tighter bounding box and longer (i.e., richer) description. The slot filler explanation generation in this case simply adds a prefix "The picture shows:" to the generated explanation — in the future, we plan to explore other more complex slot-filter templates (if necessary).

Algorithm 1 Generate XQA explanations using regions

Require: Image I, question Q, answer A, regions RS in I with descriptions DS, visual (attention) and language (LM) scoring models.
Ensure: Explanations E for Q/A, sorted by relevance.
  for all Description D in DS, for region R in RS **do**
    Compute relevance of D to Q/A using score(D, QA) defined in Equation 1, using attention and LM models.
  end for
  D' = Descriptions D sorted by decreasing score(D, QA)
  return E = Slot-filler explanations generated from D'

When the scene graph of an image just has objects and relations (and no region descriptions, like in the Visual Genome data), then we use Algorithm 2 to generate the explanations. In this algorithm, we find out the relevant objects/relations and then use a graph traversal algorithm to find a set of connected relations that form a connected component in the graph — we use the connected component to generate a "descriptive" explanation (outlined in Algorithm 3). Thus, when region descriptions are not available, we can instead use these descriptive explanations for the connected component of relations relevant to the question/answer.

4 Example Explanations

In this section, we illustrate our algorithm using the Visual Genome (VG) data, providing a few examples in which the multi-modal explanation generation algorithm performs well. For the tennis image and associated Q/A, Algorithm 1, the region-based algorithm gave the following results using the multi-modal explanation generation approach:

Algorithm 2 Generate XQA explanations using objects/relations

Require: Image I, question Q, answer A, objects or relations OS in the scene graph, visual (attention) and language (LM) models.
Ensure: Explanations E for Q/A, sorted by relevance.
  for all Object O in OS **do**
    Compute relevance of O to Q/A using score(O, QA) defined in Equation 1, using attention and LM models.
  end for
  O' = Objects O sorted by decreasing score(O, QA)
  return E = Descriptive explanations generated from O' using Algorithm 3.

Q/A: What is this game? Tennis. **Explanation**:
1. The picture shows: a tennis court
2. The picture shows: a tennis player hitting a ball
3. The picture shows: a woman hitting a tennis ball
4. The picture shows: a red and silver tennis racket
5. The picture shows: a blue and white tennis shoe

In this example, explanations 1–4 are relevant. Example 5, while mentioning the relevant concept of a tennis shoe, does not provide as satisfactory an explanation. Consider another image of a set of people at a crosswalk (Figure 6) — for this figure, we get the following results:

Figure 6: VG image of people at crosswalk.

Q/A: What is across the street? Other people. **Explanation**:
1. The picture shows: group of people across the street
2. The picture shows: buildings at the end of the street
3. The picture shows: people across street near crosswalk
4. The picture shows: a street lamp
5. The picture shows: people waiting to cross the street

As we can see in this example, explanations 1, 3 and 5 are relevant. The success of these two examples suggests that that getting relevant results in the top 5 explanations.

Our baseline method is to generate a similar explanation but without the attention heatmap guidance from the VQA model. Without the attention heatmap, the sentence generated will still be relevant to the image and question asked, but will not be explaining the decision of the model since the explanation generation is not tied to the model decision process in any way. Thus, this acts as a strong baseline since any relevant sentence to the image shouldn't be mistaken for an explanation of why a model predicted a certain answer for a question.

Algorithm 3 DFSSortedWithEmit(N): Generate descriptive explanations using objects/relations for graph rooted at N

Require: Graph G = (OS, RS), OS is set of objects (nodes), RS is set of relations (edges), maximum number of objects used in explanation kNumTermsInExplanation, language (LM) model.
Ensure: Explanations E for Q/A.
  LR = list of relations RS in decreasing order of LM model score
  LO = list of objects OS in decreasing order of LM model score
  Explanation list EL = []
  for all Relation R(O, O') in LR **do**
    if O is in LO and O is not marked **then**
      Phrase P = DFSSortedWithEmit(O)
      Add phrase P to EL
    end if
  end for
  if size(EL) $<$ kNumTermsInExplanation **then**
    for all For unmarked object O in LO: **do**
      Phrase P' = DFSSortedWithEmit(O)
      Add phrase P' to EL until size(EL) = kNumTermsInExplanation
    end for
  end if
  Create explanations E with phrases from list EL using slot-filling
  return E

We evaluate how the results for this baseline, i.e., in Equation 1 where the $attentionScore$ is not used. In that case, we expect the results to be of poorer quality. For the tennis example, the following explanations were generated using the attention map:

Figure 7: VG image of crosswalk.

Q/A: Question considered: Why is the woman holding a racket? Answer: To hit the ball. **Explanation**:
1. The picture shows: the tennis racket of the player
2. The picture shows: a red and silver tennis racket
3. The picture shows: a woman holding a tennis racket
4. The picture shows: a woman hitting a tennis ball
5. The picture shows: a woman hitting a backhand

As we can see, explanations 4 and 5 are relevant explanations. In comparison, when we don't use the attention map, we typically find that none of the explanations are quite relevant to the answer:

Q/A: Why is the woman holding a racket? To hit the ball.
Explanation:
1. The picture shows: a tennis ball
2. The picture shows: a yellow tennis ball
3. The picture shows: a small tennis ball
4. The picture shows: a red and silver tennis racket
5. The picture shows: the tennis racket of the player

To quantify the importance of the attention map in the multi-modal NL explanation generation algorithm, we performed an A/B test with human raters — the analysis of the ratings, outlined in the next section, validates our assumption that using the multi-modal approach gives us better explanations over using one modality (e.g., linguistic analysis) alone.

We next show some examples of explanations generated by using the story-like algorithm with objects and relations extracted from the scene graph without using region descriptions. In each case, we show the top explanation generated using the story-like explanation generation algorithm, as outlined in Algorithms 2 and 3:

Figure 8: VG image of office.

Q/A: Where was this picture taken? At the intersection (Figure 7)
Explanation: The picture shows crosswalk on road and in front of man, car parked on road, tree next to road, sign next to road, bike next to car, building with window, walk sign.

Q/A: Where was this picture taken? In an office (Figure 8)
Explanation: The picture shows keyboard with keys, filing cabinet with drawer, bag on desk, picture on wall, outlet on wall, pen on desk, mouse next to keyboard, filing cabinet with handle, cable on floor, cables on floor.

Q/A: Where was this picture taken? In a dining room (Figure 9)
Explanation: The picture shows chair with leg, food in bag, liquid in glass, fork on plate, bottle with logo, crumb on plate, bag rests on bowl, bar stool, cover, pan.

5 Experiments

Figure 9: VG image of dining room.

We conducted an initial quantitative user-study to demonstrate that human users were satisfied with explanations generated by our multi-modal algorithm. In this evaluation, the ratings were performed internally within our research group, however, in future experiments, we plan to use Amazon Mechanical Turk. Four internal workers rated about 220 questions — for each question, explanations were generated using both the multi-modal algorithm and the NL-only approach (as the baseline). In the latter case, visual attention was not used in the explanation generation. Overall, close to 2K explanations were rated. The authors did not know beforehand which of the explanations came from the baseline algorithm vs. our multi-modal approach, so as to not bias ratings in any way.

We used 3 metrics for evaluation in our small-scale study:

1. Explanation score: Each explanation was rated using a relevance score ranging from -5 to +5, where -5 indicates irrelevant explanation, 0 indicates redundant explanation and +5 corresponds to relevant non-redundant explanation. A negative score between 0 and -5 indicates degree of irrelevance, while a positive score between 0 and +5 indicates degree of relevance.

   Each question/answer pair was also rated, according to the degree of explainability of the question/answer, on score of 1 to 5 — 1 indicates that the question/answer pair is difficult to explain (e.g., a question/answer like "Q: What color is the shirt? A: Red"), while 5 indicates that the question is easy to explain (e.g., a question/answer like "Q: Where was this picture taken? A: On the beach").

Type	Win	Loss	Tie
Explanation score	52%	28%	20%
Position score	55%	30%	15%
Number score	54%	24%	22%

Table 1: Statistics of multi-modal approach compared to NL-only approach (Win ⇒ multi-modal wins).

When an algorithm generates explanations for a question/answer pair, we score each explanation using the relevance score, multiply that score by a position weight (so that explanations in higher positions get higher weight), and finally scale that using the question explainability (so that explanations for more explainable questions are given higher score).

2. Position of first relevant explanation: Given a set of generated explanations, this metric compares the position of the first relevant explanation.
3. Number of relevant explanations: Given a set of generated explanations, this metric measures the number of relevant explanations in the top-5 generated explanations.

In all 3 cases, the combined multi-modal algorithm outperformed the baseline NL-only approach. The results are summarized in Table 1: considering explanation score, multi-modal was better in $\approx 52\%$ cases; based on position on first relevant score, multi-modal was better in $\approx 55\%$ cases; while in $\approx 54\%$ cases, multi-modal was better than NL-only according to number of relevant explanations. These initial quantitative results demonstrate that humans are more satisfied with our explanations when those explanations came from the multi-modal algorithm that was actually tied to the inference procedure of the visual model.

6 Related Work

There is a large body of literature on automatic generation of different types of machine explanations, e.g., explanations for recommendation systems [Costa *et al.*, 2017], affordances from images [Chuang *et al.*, 2017], and robotics [Sridharan *et al.*, 2016]. Explanation generation has also been explored in the areas of planning [Fox *et al.*, 2017], interactive model debugging [Kulesza *et al.*, 2015], autonomous systems [Langley *et al.*, 2017], mobile robotics [Rosenthal *et al.*, 2016], expert systems [Swartout *et al.*, 1991], tactical behavior modeling [van Lent *et al.*, 2004], etc.

In this paper, we focus on NL explanations for the visual question answering task, where previous work has been done by Hendricks et al. [Hendricks *et al.*, 2016] and others. A closely-related work to our approach is VQA-X, a method for generating explanation datasets [Park *et al.*, 2017], where the authors propose a multi-modal methodology for simultaneously generating visual and textual explanations. Another related work for generating NL explanations using a multi-modal approach [Park *et al.*, 2018] qualitatively show cases where visual explanation is more insightful than textual explanation (and vice versa), demonstrating that multi-modal explanation models offer significant benefits over uni-modal approaches. Note that both these approaches rely on getting a large corpus of explanations from human annotators for training the models. In our proposed method, we don't need manually generated explanation data, we use already available annotations from scene graphs only.

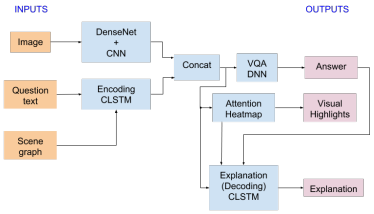


Figure 10: Future Explanation Generation Pipeline.

7 Conclusions and Future Work

In this paper, we presented a multi-modal approach for generating natural language explanations for the visual question answering task, using both visual and linguistic modalities, without collecting any additional data. We also showed empirically how the multi-modal approach gives better explanations than using one modality (e.g., linguistic analysis) alone.

We also plan to look into training effective models for learning to generate explanations while predicting answers given an image, context about the image, and a question. Image context may involve scene graphs and we can use Contextual LSTMs (CLSTMs) [Ghosh *et al.*, 2016] to encode this additional information. The main benefit of training a pipeline to do explanation generation (as suggested in Figure 10), instead of using hard-coded algorithms (as suggested in Figure 5), is that models have higher flexibility to learn complicated common-sense semantic information, if that is necessary to generate a satisfactory explanation.

We would also like to explore some other improvements to our system, namely: (1) Infer super-categories of relations and objects (e.g., obtained from hypernyms in wordnet), add them as a preface to the explanation list. (2) Explore the use of embeddings, e.g., skip-thought vectors [Kiros *et al.*, 2015] to find the similarity of entity descriptions (e.g., object attributes, relation phrases or region descriptions) with the salient parts of the question/answer text. (3) Conducting a study to check if our textual explanations help humans predict VQA accuracy for a given image-question pair.

Acknowledgments: The authors would like to thank Kamran Alipour and Jurgen Schulze for setting up a web demo for our VQA-Interface, and Dr. Ajay Divakaran and Dr. Yi Yao for valuable feedback and suggestions. We would like to thank Dr. Patrick Lincoln for his valuable feedback and support. This research was developed with funding from the Defense Advanced Research Projects Agency (DARPA), under the Explainable Artificial Intelligence (XAI) program. The views, opinions and/or findings expressed are those of the author and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

References

[Antol *et al.*, 2015] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. VQA: Visual Question Answering. In *International Conference on Computer Vision (ICCV)*, 2015.

[Chuang *et al.*, 2017] Ching-Yao Chuang, Jiaman Li, Antonio Torralba, and Sanja Fidler. Learning to act properly: Predicting and explaining affordances from images. *CoRR*, abs/1712.07576, 2017.

[Costa *et al.*, 2017] Felipe Costa, Sixun Ouyang, Peter Dolog, and Aonghus Lawlor. Automatic generation of natural language explanations. *CoRR*, abs/1707.01561, 2017.

[Das *et al.*, 2016] Abhishek Das, Satwik Kottur, Khushi Gupta, Avi Singh, Deshraj Yadav, José M. F. Moura, Devi Parikh, and Dhruv Batra. Visual dialog. *CoRR*, abs/1611.08669, 2016.

[Fox *et al.*, 2017] Maria Fox, Derek Long, and Daniele Magazzeni. Explainable planning. *CoRR*, abs/1709.10256, 2017.

[Ghosh *et al.*, 2016] Shalini Ghosh, Oriol Vinyals, Brian Strope, Scott Roy, Tom Dean, and Larry P. Heck. Contextual LSTM (CLSTM) models for large scale NLP tasks. *CoRR*, abs/1602.06291, 2016.

[He *et al.*, 2015] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[Hendricks *et al.*, 2016] Lisa Anne Hendricks, Zeynep Akata, Marcus Rohrbach, Jeff Donahue, Bernt Schiele, and Trevor Darrell. Generating visual explanations. *CoRR*, abs/1603.08507, 2016.

[Józefowicz *et al.*, 2016] Rafal Józefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. Exploring the limits of language modeling. *CoRR*, abs/1602.02410, 2016.

[Kazemi and Elqursh, 2017] Vahid Kazemi and Ali Elqursh. Show, ask, attend, and answer: A strong baseline for visual question answering. *arXiv preprint arXiv:1704.03162*, 2017.

[Kiros *et al.*, 2015] Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. Skip-thought vectors. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'15, 2015.

[Krishna *et al.*, 2016] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, Michael Bernstein, and Li Fei-Fei. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *CoRR*, abs/1602.07332, 2016.

[Kulesza *et al.*, 2015] Todd Kulesza, Margaret Burnett, Weng-Keen Wong, and Simone Stumpf. Principles of explanatory debugging to personalize interactive machine learning. In *Proceedings of the 20th International Conference on Intelligent User Interfaces*, 2015.

[Langley *et al.*, 2017] Pat Langley, Ben Meadows, Mohan Sridharan, and Dongkyu Choi. Explainable agency for intelligent autonomous systems. In *AAAI*, 2017.

[Lu *et al.*, 2016] Cewu Lu, Ranjay Krishna, Michael S. Bernstein, and Fei-Fei Li. Visual relationship detection with language priors. *CoRR*, abs/1608.00187, 2016.

[Park *et al.*, 2017] Dong Huk Park, Lisa Anne Hendricks, Zeynep Akata, Anna Rohrbach, Bernt Schiele, Trevor Darrell, and Marcus Rohrbach. Attentive explanations: Justifying decisions and pointing to the evidence (extended abstract). *CoRR*, abs/1711.07373, 2017.

[Park *et al.*, 2018] Dong Huk Park, Lisa Anne Hendricks, Zeynep Akata, Anna Rohrbach, Bernt Schiele, Trevor Darrell, and Marcus Rohrbach. Multimodal explanations: Justifying decisions and pointing to the evidence. *CoRR*, abs/1802.08129, 2018.

[Pennington *et al.*, 2014] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[Ray *et al.*, 2016] Arijit Ray, Gordon Christie, Mohit Bansal, Dhruv Batra, and Devi Parikh. Question relevance in vqa: identifying non-visual and false-premise questions. *arXiv preprint arXiv:1606.06622*, 2016.

[Ray, 2017] Arijit Ray. *The Art of Deep Connection-Towards Natural and Pragmatic Conversational Agent Interactions*. PhD thesis, Virginia Tech, 2017.

[Rosenthal *et al.*, 2016] Stephanie Rosenthal, Sai P. Selvaraj, and Manuela Veloso. Verbalization: Narration of autonomous robot experience. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI)*, 2016.

[Sridharan *et al.*, 2016] Mohan Sridharan, Ben Meadows, and Zenon Colaco. A tale of many explanations: Towards an explanation generation system for robots. In *Proceedings of the 31st Annual ACM Symposium on Applied Computing*, 2016.

[Swartout *et al.*, 1991] W. Swartout, C. Paris, and J. Moore. Explanations in knowledge systems: design for explainable expert systems. *IEEE Expert*, 6(3), 1991.

[Teney *et al.*, 2017] Damien Teney, Peter Anderson, Xiaodong He, and Anton van den Hengel. Tips and tricks for visual question answering: Learnings from the 2017 challenge. *CoRR*, abs/1708.02711, 2017.

[van Lent *et al.*, 2004] Michael van Lent, William Fisher, and Michael Mancuso. An explainable artificial intelligence system for small-unit tactical behavior. In *Proceedings of the 16th Conference on Innovative Applications of Artifical Intelligence*, IAAI'04, 2004.

Interpretable self-labeling semi-supervised classifier

Isel Grau[1], **Dipankar Sengupta**[1], **Maria M. Garcia Lorenzo**[2], **Ann Nowe**[1]
[1] Artificial Intelligence Laboratory, Vrije Universiteit Brussel, Belgium
[2] Computer Science Department, Universidad Central de Las Villas, Cuba
igraugar@vub.be

Abstract

Semi-supervised classification refers to a type of pattern classification problem involving both labeled and unlabeled data, where the number of labeled instances is often significantly smaller compared to the number of unlabeled ones. Although there exist several semi-supervised classifiers with high performance over different tasks, most of them are complex models that do not allow explaining the obtained outcome, thus behaving like black boxes. In this paper, we perform a critical analysis of the interpretability of state-of-the-art semi-supervised classification approaches. In addition, we present a self-labeling grey-box classifier that uses a black-box to estimate the missing class labels and an interpretable white-box to make the actual predictions. The main contribution of this model relies on its transparency while also being able to outperform most state-of-the-art semi-supervised classifiers.

1 Introduction

Given a classification problem where there exist both labeled and unlabeled data, a common approach is to use of semi-supervised learning techniques. This scenario often occurs when obtaining the examples is relatively easy, but the process of labeling them is costly either in terms of time, money or effort. The objective of semi-supervised classification (SSC) is to outperform the results obtained with a supervised technique based on the labeled data only, by using the extra unlabeled data in a clever way. This approach has been applied to real-world scenarios such as image classification [Aydav and Minz, 2016], web spam classification [Fang *et al.*, 2015], sentiment analysis [Khan *et al.*, 2016], speaker identification [Fazakis *et al.*, 2015] as well as bioinformatics or medical applications [Triguero *et al.*, 2014].

Semi-supervised classification approaches primarily vary in the assumptions about the distribution of the unlabeled data [Triguero *et al.*, 2015]. A wide range of SSC algorithms have been proposed, including Generative Mixture Models [Fujino *et al.*, 2008], transductive Support Vector Machines [Joachims, 1999] and Graph-based methods [Blum and Chawla, 2001]. Another group of SSC methods includes self-labeling models such as Co-training, Self-training and their variants [Triguero *et al.*, 2015]. Self-labeling techniques use one or several base classifiers for learning from the labeled data and then predicting the decision classes for unlabeled instances.

While existing SSC algorithms are capable of achieving good prediction rates, another challenge comes to light: experts not only need accurate predictors but also understandable ones, which can be interpretable to some extent. Regrettably, most successful SSC algorithms comprise complex structures involving collaboration between several classifiers. This results in models that are really difficult to understand and lack transparency in their functioning. In general, more interpretable supervised classifiers (e.g. Decision Trees or Linear models) have the advantage of being explainable to some extent, but generally have poorer performance in terms of prediction accuracy when compared with black-box models, such as Random Forests. This issue becomes even more apparent in semi-supervised settings where the amount of labeled data is rather limited.

In this paper, we present a critical review of the main approaches for semi-supervised classification from an interpretability point of view. In addition, we propose a simple yet highly effective SSC classifier —termed *self-labeling grey-box*— which combines the performance of a black-box classifier (such as Random Forest) with a more interpretable Decision Tree in a self-labeling scheme. As a first component, the black-box estimates the decision class for unlabeled instances increasing the amount of training data. As a second component, we build a white-box classifier from the enlarged dataset that allows explaining the predictions. Using the enlarged dataset during the training process, combined with a weighting procedure, results in a white-box model with improved prediction rates. Numerical experiments using 14 benchmark datasets show that the proposed semi-supervised classifier has a good balance between accuracy and transparency.

The rest of this paper is structured as follows. Section 2 formalizes the semi-supervised classification problem and analyzes the main approaches for SSC and their interpretability. Section 3 describes the proposed *self-labeling grey-box* model. Section 4 reports the experiments performed and a discussion covering both performance and interpretability of the model. Section 5 formalizes the concluding remarks and suggests future research lines.

2 Semi-supervised Classification

The SSC problem refers to a type of *weak supervised learning* where the goal is to increase the performance of a supervised classifier using extra unlabeled data. Let us suppose a set of M instances $L = \{l_1, l_2, \ldots, l_M\}$ with corresponding class labels $Y = \{y_1, y_2, \ldots, y_M\}$, and a set of N unlabeled instances $U = \{u_{M+1}, u_{M+2}, \ldots, u_{M+N}\}$, with $N >> M$. In this context, the performance of a classification model could be evaluated in two different settings: (1) the transductive learning, which only attempts to predict the labels for the given unlabeled instances in U; or (2) the inductive learning, which tries to infer a mapping $g : L \cup U \rightarrow Y$ in order to predict the proper class label of any possible instance associated to the classification problem. We focus on the more challenging inductive learning, since it allows to exploit the learned classifier in real-world scenarios.

2.1 Interpretability in SSC

In this section, we review the main state-of-the-art approaches for semi-supervised classification from the interpretability point of view. We consider the interpretability as the inherent model transparency, as described in [Lipton, 2016].

Existing algorithms for SSC problems can be classified into four large groups, based on their assumptions about the distribution of unlabeled data. A first group, Generative Mixture Models [Fujino *et al.*, 2008], assumes that the distribution of unlabeled examples follows an identifiable statistical mixture distribution similar to the labeled examples. The inference involves the estimation of a mixture distribution $p(x|y)$ (e.g. Gaussian mixture models), which could be interpretable at a very high level of abstraction if the representation space of the problem at hand is not too complex. However the unlabeled data could actually hurt if the generative model is wrong, therefore this technique should be used when there is evidence from the knowledge domain that supports the chosen distribution.

A second approach is to assume low-density separation of the data, i.e. the decision boundary should lie in a low-density region. The most common method for maximizing the margin for unlabeled as well as labeled points is called transductive Support Vector Machine (tSVM) [Joachims, 1999]. Given that the complexity of the optimization problem increases in the semi-supervised setting, the computational cost of this technique is very high. In addition, although supervised SVM is a powerfull technique with a strong mathematical framework for building classifiers, from the interpretability point of view it works as a black-box. A related third group consist of Graph-based methods [Blum and Chawla, 2001] assumes that the high-dimensional data lie on a low-dimensional manifold [Chapelle *et al.*, 2010]. Graph-based methods then represent the data space as a graph, i.e., if two instances are strongly connected, then they likely belong to the same class. They estimate a continuous function closely enough to the label values, with the ultimate goal of propagating labels between similar instances. This approach could be interpretable to some extent by analyzing the obtained graph. Regrettably, this technique is only oriented to the transductive setting and cannot be used for predicting new unlabeled instances without re-learning the classifier.

The fourth group consist of the self-labeling techniques, which allow solving SSC problems in both *transductive* as well as *inductive* settings. These techniques use a base classifier (or a combination) in order to predict the labels of unlabeled data. In a second step, these methods use the self-labeled data for enhancing their learning process. Therefore, they assume that the predictions obtained in the self-training process are correct. Self-labeling techniques are easy to implement and apply to almost all existing classifiers. An exhaustive experiment conducted in [Triguero *et al.*, 2015] shows that CoTraining using Support Vector Machines as a base classifier [Hady and Schwenker, 2008], TriTraining using C4.5 decision tree [Zhou and Li, 2005], CoBagging using C4.5 [Hady and Schwenker, 2008] and Democratic Co-learning [Zhou and Goldman, 2004], are the best performing self-labeling classifiers reported in literature.

Although these techniques often compute higher prediction rates when compared to other approaches [Triguero *et al.*, 2015], the outcome of these methods is difficult to understand at the whole model level. Most of the algorithms for SSC problems behave as black-boxes as their base classifiers hardly elucidate why they produce a particular decision for a new observation. We consider the interpretability of SSC techniques to be related to the transparency of its building blocks, as well as the complexity of the final model. For example, a self-training scheme producing a (relatively) simple tree structure as the final classifier could be considered a transparent model. More complex schema such as the multi-classifier models (e.g. Tri-training, Co-Bagging, or Co-Forest) or multi-view ones (e.g. Co-training) are also more difficult to interpret at a model level due to the collaborative nature of the algorithms and the complexity of the resulting ensemble classifier.

Several post-hoc approaches for explaining the inference of supervised classifiers have been also proposed in literature. Post-hoc interpretations often do not explain how a model works, but they provide useful information for explaining the inference using natural language, visualization, etc [Lipton, 2016]. This can be seen as another attempt to interpret trained black-boxes without sacrificing predictive performance, but does not offer insight into the knowledge domain of the modeled problem.

3 Self-labeling Grey-box

In order to build an SSC model with balance between accuracy and interpretability, we present a simple yet effective ensemble scheme called *self-labeling grey-box* (SlGb). In terms of interpretability, a grey-box model is referred to as the combination of a black-box model with a white-box one. Black-boxes are normally more accurate techniques that learn exclusively from historical data but are not easily understandable at a model level (e.g. random forest, support vector machines, (deep) neural networks). Whereas white-boxes refer to models which are constructed based on laws or principles of the problem domain, or those who are built from historical data but their structure allows explanations (e.g. rule-based mod-

els, decision trees), since pure white-box rarely exists [Nelles, 2013]. Our SlGb uses a black-box classifier to predict the decision class of the unlabeled instances, while a white-box component is used to build an interpretable predictive model (e.g., a rule-based approach), based on the whole instance set. The aim is to outperform the performance of the base white-box component using only the labeled data, whilst finding a good balance between performance and interpretability.

In a first learning step, a black box model is trained using the available labeled dataset (L, Y). Once the supervised learning is completed, the black box has learned a function $f : L \rightarrow Y$, with $f \in F$, being F the hypothesis space. More precisely we consider hypotheses based on scoring functions. Let $h : L \times Y \rightarrow [0, 1]$ be a scoring function such that $f(x) = argmax_{y \in Y}\{h(x, y)\}, x \in L$. Then, the black box is used for predicting the labels y_k, for unlabeled instances $u_k \in U$ as $y_k = f(u_k)$, $k = \overline{M+1, M+N}$, adding a self-labeling character to the model. From this learning stage we obtain an enlarged training set $(L \cup U, Y \cup Y^u)$ comprising the original labeled instances and the extra labeled ones.

In the second learning step, the enlarged training set $(L \cup U, Y \cup Y^u)$ obtained in the previous step is used to train a white box model. This implies that we obtain a function $g : (L \cup U) \rightarrow (Y \cup Y^u)$ resulting in a classifier with presumably better generalization ability compared to being trained on the labeled data only. Figure 1 illustrates the whole process in detail.

An important point of concern is that in self-labeling techniques the mistakes in the classification can reinforce themselves if no amending procedure is used during self-training. In addition, imbalance datasets are very common in semi-supervised scenarios, due to the data usually comes from different sources. In order to deal with these problems, our scheme additionally incorporates a procedure for weighting the instances. For labeled instances, the weight is computed as the proportion of instances belonging to that class, giving a higher importance to instances belonging to the minority class. In the case of unlabeled instances u_k, the weights are assigned after being labeled by the black-box classifier and they express the confidence degree associated to the self-labeling process. More explicitly, the weight is equal to the learned scoring function $h(u_k, y_k)$ which expresses the probability (or confidence) of being correctly assigned to the particular class y_k. Equations 1 and 2 formalize the weighting process for labeled and unlabeled instances, respectively, as follows:

$$w_{l_i} = \frac{|L_{min}|}{|L_i|} \tag{1}$$

$$w_{u_k} = h(u_k, y_k) \tag{2}$$

where $L_{min} \subset L$ is the subset of instances of the minority class y_{min} in the labeled train set and $L_i \subset L$ is the subset of instances with class y_i in the set of labeled instances. In general, the proposed SlGb model is only based on the general assumption of SSC methods: the distribution of unlabeled instances helps elucidate the distribution of all examples. The simplicity of the proposed scheme leads to maintain (to some extent) the inherent interpretability features of the chosen white-box component in the final grey-box classifier.

4 Experiments

In this section, we evaluate the predictive ability of the proposed SlGb classifier using standard benchmark datasets for SSC problems. We compare our scheme against the best-performing self-labeling algorithms reported in [Triguero *et al.*, 2015] using benchmark datasets. We perform an analysis of the results and discuss the interpretability of the resulting classifier.

4.1 Benchmark datasets, base classifiers and parameter settings

In order to perform the numerical simulations, we use a publicly available benchmark of 14 datasets for SCC [Triguero *et al.*, 2015] comprising different characteristics: the number of instances ranges from 100 to 1500, the number of attributes from 3 to 60, and the number of decision classes from 2 to 11 (see Table 1). These datasets are partitioned in training and test sets using a standard 10-fold cross-validation, and each training set consist of labeled and unlabeled instances. The set of unlabeled instances is obtained by performing a random selection without replacement and neglecting the class label of such instances. Only the 10% of the training set retains the label information (10% labeling ratio)[1].

Table 1: Datasets used in the experiments.

Dataset	Instances	Attributes	Classes
appendicitis	106	7	2
dermatology	366	33	6
haberman	306	3	2
heart	270	13	2
hepatitis	155	19	2
housevotes	435	16	2
iris	150	4	3
monks	432	6	2
sonar	208	60	2
spectheart	267	44	2
tae	151	5	3
vehicle	846	18	4
vowel	990	13	11
yeast	1,484	8	10

As base classifiers we employ Random Forests [Breiman, 2001] as black-box component and C4.5 Decision Trees [Quinlan, 1993] as white-box component. The Random Forest generates a bagging of 100 Random Trees and selected the number of random attributes as $\log_2(numberofattributes) + 1$ for each tree. The scoring function $h(u_k, y_k)$ is computed as the fraction of random trees in the forest that classify the instance u_k as label y_k,

[1]These datasets are publicly available at `www.keel.es`. The fold partitions are provided, therefore allowing to use the exact same experimental setup for comparison.

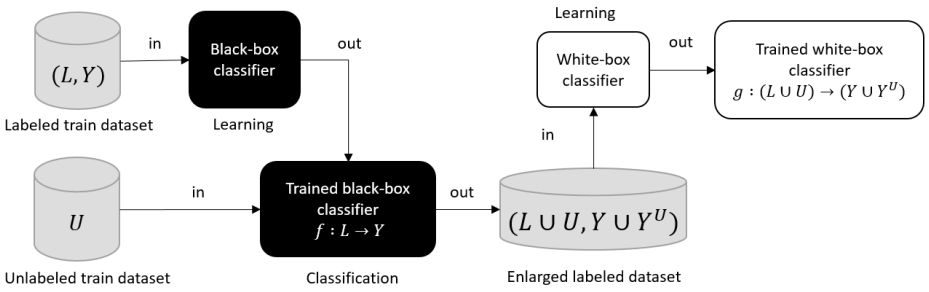


Figure 1: Data flow in the learning process of the Self-labeling Grey-box.

with respect to the total number of trees. The C4.5 Decision Tree uses a confidence level of 0.25 and prunning[2].

4.2 Comparison to other self-labeling models

The following experiment compares the predictive capability of the self-labeling grey-box using random forest and decision tree (SlGb(RFDT)) against the 4 best self-labeling models reported in [Triguero *et al.*, 2015]: Co-training using SMO [Hady and Schwenker, 2008] (CT(SMO)), Tri-training using C45 [Zhou and Li, 2005] (TT(C45)), Co-Bagging using C45 [Hady and Schwenker, 2008] (CB(C45)) and Democratic Co-learning [Zhou and Goldman, 2004] (DCT). The Table 2 reports the accuracy and Cohen's kappa obtained by each classifier across all datasets.

In order to provide a rigorous statistical analysis, we compute the Friedman two-way analysis of variances by ranks [Friedman, 1937]. The test suggests rejecting the null hypotheses based on a confidence interval of 95% with a p-value$= 0.004 < 0.05$. Meaning there is an indication that there are significant differences between at least two algorithms. Figure 2 shows the mean ranks (the higher, the better) where the RF-DT grey-box is the best-performing classifier across all datasets included in the study.

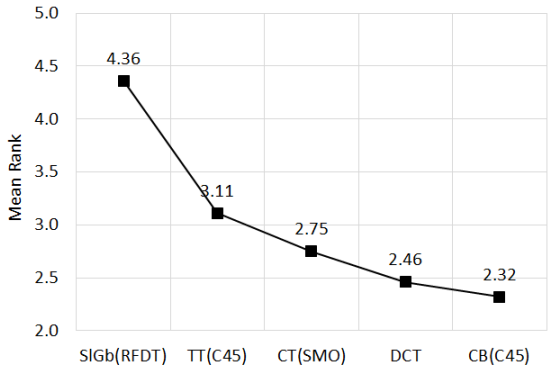


Figure 2: Mean rank values computed by the Friedman test. SlGb(RFDT) is the best performing algorithm.

The next section focusses on determining whether the superiority of the SlGb(RFDT) classifier is responsible for the significant difference reported by the Friedman test or not. To do so, we adopt the Wilcoxon signed rank test [Wilcoxon, 1945]. Table 3 reports the p-value computed by the Wilcoxon test and the and the positive ($R+$) and negative ($R-$) ranks associated with each pairwise comparison using SlGb(RFDT) as the control method. The null hypothesis states that there is no significant difference between the performance of each pair of algorithms in the comparison, taking our SlGb(RFDT) as the control algorithm.

From the table's results we can draw following key conclusions. The results show that SlGb(RFDT) is the best-performing classifier, with significant differences observed when compared against DCT and CT(SMO) based on a confidence interval of 95%, and moderately significant differences when compared to TT(C45) and CB(C45) based on a confidence interval of 90%. Although our main goal is not only outperforming the SSC algorithms in terms of classification performance, the analysis reported above supports our claim that we obtain a favorable balance between performance and interpretability by using the proposed grey-box classifier for solving SSC problems.

4.3 Interpretability analysis

In contrast with performance measures, there is a clear lack of measures for interpretability in the literature. Therefore, it is difficult to compare two algorithms of different nature with respect to their interpretability. We chose to show the average size of the decision tree structure generated by the SlGb(RFDT) classifier. We use the average number of rules (leaves of the tree) and antecedents (levels of the tree) as a measure of the simplicity of the tree. We compare against the structure generated by a supervised decision tree, learned with only the labeled data available and using the same parameters. Next Table 4 shows the results obtained.

In general, it can be observed that the decision tree generated by the semi-supervised technique is more complex than the supervised decision tree on the labeled data only. For example, on the well-known Iris dataset, the generated tree using DT has on average (across each fold of the cross-validation) 5 rules and 3 levels; while the DT component in the grey-box scheme has in average 5.2 rules and 3.1 levels. However, our grey-box achieves a kappa value of 0.91, while the DT using the same amount of labeled data only reaches 0.73. The increase in the complexity of the tree is an expected consequence of providing more data to the DT in the grey-box

[2]The SlGb algorithms are implemented within WEKA platform [Witten *et al.*, 2017].

Table 2: Average accuracy and Cohen's Kappa achieved by the algorithms over 10-fold cross-validation. Best performing algorithms according to Kappa value are highlighted in bold.

	SlGb(RFDT)		CT(SMO)		DCT		TT(C45)		CB(C45)	
Dataset	Acc.	Kappa	Acc.	Kappa	Acc.	Kappa	Acc.	Kappa	Acc.	Kappa
appendicitis	0.82	**0.45**	0.79	0.26	0.82	0.12	0.80	0.25	0.80	0.25
dermatology	0.89	**0.87**	0.77	0.71	0.88	0.84	0.88	0.85	0.88	0.84
haberman	0.64	**0.22**	0.59	0.06	0.72	0.07	0.71	0.12	0.71	0.17
heart	0.77	0.54	0.74	0.47	0.80	**0.59**	0.71	0.42	0.70	0.39
hepatitis	0.79	**0.19**	0.82	0.00	0.83	0.00	0.83	0.00	0.83	0.00
housevotes	0.92	**0.85**	0.83	0.64	0.89	0.78	0.92	0.83	0.92	0.84
iris	0.94	0.91	0.95	**0.92**	0.91	0.87	0.73	0.59	0.80	0.70
monk-2	0.96	0.92	0.81	0.63	0.91	0.82	0.97	**0.93**	0.97	**0.93**
sonar	0.69	0.37	0.60	0.15	0.60	0.17	0.70	**0.39**	0.70	**0.39**
spectfheart	0.76	**0.36**	0.78	0.33	0.74	**0.36**	0.76	0.20	0.76	0.18
tae	0.35	0.03	0.33	0.00	0.38	0.07	0.46	**0.19**	0.42	0.13
vehicle	0.64	**0.52**	0.61	0.48	0.50	0.34	0.62	0.49	0.60	0.47
vowel	0.51	**0.47**	0.46	0.40	0.42	0.36	0.45	0.40	0.46	0.40
yeast	0.53	**0.40**	0.48	0.33	0.49	0.34	0.49	0.34	0.48	0.32

Table 3: Obtained p-values, positive and negative ranks, using SlGb(RFDT) classifier as control method.

SlGb(RFDT) compared with	p**-value**	$R+$	$R-$
TT(C45)	0.19	3	11
CB(C45)	0.16	3	11
DCT	0.006	2	12
CT(SMO)	0.001	1	13

Table 4: Average growth in the decision tree structure generated by SlGb(RFDT) compared to supervised C45

	SlGb(RFDT)		supervised C45	
Dataset	# rules	# levels	# rules	# levels
appendicitis	3.4	2.2	1.4	1.2
dermatology	13.2	7.1	10.2	5.6
haberman	11.6	6.3	3	2
heart	14.8	7.9	5.6	3.3
hepatitis	4.4	2.7	1.2	1.1
housevotes	7.2	4.1	3.2	2.1
iris	5.2	3.1	5	3
monk-2	6.6	3.8	5	3
sonar	11	6	3.8	2.4
spectfheart	15.2	8.1	5.6	3.3
tae	12.6	6.8	5.6	3.3
vehicle	52	26.5	24.4	12.7
vowel	78.2	39.6	40	20.5
yeast	62	31.5	41	21

scheme, however, this comes with a very significant increase in performance.

5 Conclusions

Although there are several methods for addressing semi-supervised classification problems with good performance, most of them are not transparent at the model level. The complexity of the involved base classifier(s) and the ensemble scheme of some of these techniques do not allow to readily present a model to the final user which can be easily understood.

In this paper, we present a semi-supervised classifier which uses a self-learning approach combining two base classifiers: a black-box for the prediction of unlabeled instances and a more interpretable white-box for constructing the final model. We tested our self-labeling grey-box using random forest as the black-box and C4.5 decision trees as the white-box. The experiments show that our algorithm outperforms the top self-labeling techniques for several benchmark datasets.

When comparing the structure of the decision tree generated by the self-labeling grey-box with the one obtained from the equivalent supervised classification problem, we observed that the structure of the tree increases. This is to be expected and is a consequence of providing more data to the decision tree, but allows us to significantly increase the performance of the classifier without completely sacrificing the transparency of the final model, as in state-of-the-art approaches.

A future research direction includes to address more challenging application problems while evaluating the interpretability of the classifier, in collaboration with experts in the application field. We have taken the first steps into comparing the general knowledge obtained from the classifier with the output of instance based post-hoc explanation methods. For the particular case of decision trees used as interpretable models, more work is needed in order to maintain an acceptable complexity of the final classifier.

References

[Aydav and Minz, 2016] Prem Shankar Singh Aydav and Sonjharia Minz. Semi-supervised fcm and svm in co-training framework for the classification of hyperspectral images. In *Intelligent Systems Technologies and Applications 2016*, pages 119–129. Springer International Publishing, 2016.

[Blum and Chawla, 2001] Avrim Blum and Shuchi Chawla. Learning from labeled and unlabeled data using graph mincuts. In *Proceedings of the Eighteenth International*

Conference on Machine Learning, ICML '01, pages 19–26, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[Breiman, 2001] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[Chapelle *et al.*, 2010] Olivier Chapelle, Bernhard Schlkopf, and Alexander Zien. *Semi-Supervised Learning*. The MIT Press, 1st edition, 2010.

[Fang *et al.*, 2015] Xiaonan Fang, Yanyan Tan, Xiyuan Zheng, Huaxiang Zhang, and Shuang Zhou. Imbalanced web spam classification using self-labeled techniques and multi-classifier models. In *Proceedings of the 8th International Conference on Knowledge Science, Engineering and Management*, pages 663–668. Springer International Publishing, 2015.

[Fazakis *et al.*, 2015] Nikos Fazakis, Stamatis Karlos, Sotiris Kotsiantis, and Kyriakos Sgarbas. Speaker identification using semi-supervised learning. In *International Conference on Speech and Computer*, pages 389–396. Springer, 2015.

[Friedman, 1937] Milton Friedman. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, 32(200):675–701, 1937.

[Fujino *et al.*, 2008] Akinori Fujino, Naonori Ueda, and Kazumi Saito. Semisupervised learning for a hybrid generative/discriminative classifier based on the maximum entropy principle. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(3):424–437, 2008.

[Hady and Schwenker, 2008] Mohamed Farouk Abdel Hady and Friedhelm Schwenker. Co-training by committee: a new semi-supervised learning framework. In *IEEE International Conference on Data Mining Workshops*, ICDMW '08, pages 563–572. IEEE, 2008.

[Joachims, 1999] Thorsten Joachims. Transductive inference for text classification using support vector machines. In *Proceedings of the Sixteenth International Conference on Machine Learning*, ICML '99, pages 200–209, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[Khan *et al.*, 2016] Farhan Hassan Khan, Usman Qamar, and Saba Bashir. A semi-supervised approach to sentiment analysis using revised sentiment strength based on sentiwordnet. *Knowledge and Information Systems*, pages 1–22, 2016.

[Lipton, 2016] Zachary C. Lipton. The mythos of model interpretability. In *2016 ICML Workshop on Human Interpretability in Machine Learning*, WHI 2016, pages 96–100, 2016.

[Nelles, 2013] Oliver Nelles. *Nonlinear system identification: from classical approaches to neural networks and fuzzy models*. Springer Science & Business Media, 2013.

[Quinlan, 1993] J Ross Quinlan. C4. 5: Programming for machine learning. *Morgan Kauffmann*, page 38, 1993.

[Triguero *et al.*, 2014] Isaac Triguero, Salvador García, Francisco Herrera, and Yvan Saeys. Improving disease prediction using unlabeled and synthetic samples. In *Proceedings of the Benelux Bioinformatics Conference 2014*, page 72, 2014.

[Triguero *et al.*, 2015] Isaac Triguero, Salvador García, and Francisco Herrera. Self-labeled techniques for semi-supervised learning: taxonomy, software and empirical study. *Knowledge and Information Systems*, 42(2):245–284, Feb 2015.

[Wilcoxon, 1945] F. Wilcoxon. Individual comparisons by ranking methods. *Biometrics*, 1:80–83, 1945.

[Witten *et al.*, 2017] Ian H. Witten, Eibe Frank, Mark A. Hall, and Christopher J. Pal. *Data Mining Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Publishers Inc., fourth edition, 2017.

[Zhou and Goldman, 2004] Yan Zhou and Sally Goldman. Democratic co-learning. In *Proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence*, ICTAI 2004, pages 594–602. IEEE, 2004.

[Zhou and Li, 2005] Zhi-Hua Zhou and Ming Li. Tri-training: Exploiting unlabeled data using three classifiers. *IEEE Transactions on Knowledge and Data Engineering*, 17(11):1529–1541, 2005.

How explainable plans can make planning faster*

Antoine Gréa and **Laetitia Matignon** and **Samir Aknine**

Abstract

In recent years the ubiquity of artificial intelligence raised concerns among the uninitiated. The misunderstanding is further increased since most advances do not have explainable results. For automated planning, the research often targets speed, quality, or expressivity. Most existing solutions focus on one criteria while not addressing the others. However, human-related applications require a complex combination of all those criteria at different levels. We present a new method to compromise on these aspects while staying explainable. We aim to leave the range of potential applications as wide as possible but our main targets are human intent recognition and assistive robotics. We propose the HEART planner, a real-time decompositional planner based on a hierarchical version of Partial Order Causal Link (POCL). It cyclically explores the plan space while making sure that intermediary high level plans are valid and will return them as approximate solutions when interrupted. These plans are proven to be a guarantee of solvability. This paper aims to evaluate that process and its results compared to classical approaches in terms of efficiency and quality.

Introduction

Since the early days of automated planning, a wide variety of approaches have been considered to solve diverse types of problems. They all range in expressivity, speed, and reliability but often aim to excel in one of these domains. This leads to a polarization of the solutions toward more specialized methods to tackle each problem. All of these approaches have been compared and discussed extensively in the books of Ghallab et al. [2004; 2016].

Partially ordered approaches are popular for their least commitment aspect, flexibility and ability to modify plans using refinement operations [Weld 1994]. These approaches are often used in applications in robotics and multi-agent planning [Lemai and Ingrand 2004; Dvorak *et al.* 2014]. One of the most flexible partially ordered approaches is called *Partial Order Causal Link planning (POCL)* [Young and Moore 1994]. It works by refining partial plans consisting of steps and causal links into a solution by solving all flaws compromising the validity of the plan.

Another approach is *Hierarchical Task Networks (HTN)* [Sacerdoti 1974] that is meant to tackle the problem using composite actions in order to define hierarchical tasks within the plan. Hierarchical domains are often considered easier to conceive and maintain by experts mainly because they seem closer to the way we think about these problems [Sacerdoti 1975].

In our work, we aim combining HTN planning and POCL planning in such a way as to generate intermediary high level plans during the planning process. Combining these two approaches is not new [Young and Moore 1994; Kambhampati *et al.* 1998; Biundo and Schattenberg 2001]. Our work is based on *Hierarchical Partial Order Planning (HiPOP)* by Bechon *et al.* [2014]. The idea is to expand the classical POCL algorithm with new flaws in order to make it compatible with HTN problems and allowing the production of abstract plans. To do so, we present an upgraded planning framework that aims to simplify and factorize all notions to their minimal forms. We also propose some domain compilation techniques to reduce the work of the expert conceiving the domain.

In all these works, only the final solution to the input problem is considered. That is a good approach to classical planning except when no solutions can be found (or when none exists). Our work focuses on the case when the solution could not be found in time or when high level explanations are preferable to the complete implementation detail of the plan. This is done by focusing the planning effort toward finding intermediary abstract plans along the path to the complete solution.

In the rest of the paper, we detail how the HiErarchical Abstraction for Real-Time (HEART) planner creates abstract intermediary plans that can be used for various applications. First, we discuss the motivations and related works to detail the choices behind our design process. Then we present the way we modeled our own planning framework fitting our needs and then we explain our method and prove its properties to finally discuss the experimental results.

1 Motivations and Potential Applications

Several reasons can cause a problem to be unsolvable. The most obvious case is that no solution exists that meets the requirements of the problem. This has already been addressed by Göbelbecker *et al.* [2010] where "excuses" are being investigated as potential explanations for when a problem has no solution.

*Univ Lyon, Université Lyon 1, CNRS, LIRIS, UMR5205, F-69621, LYON, France (first.lastname@liris.cnrs.fr)

Our approach deals with the cases of when the problem is too difficult to solve within tight time constraints. For example, in robotics, systems often need to be run within refresh rates of several Hertz giving the process only fractions of a second to give an updated result. Since planning is at least EXPSPACE-hard for HTN using complex representation [Erol *et al.* 1994], computing only the first plan level of a hierarchical domain is much easier in relation to the complete problem.

While abstract plans are not complete solutions, they still display a useful set of properties for various applications. The most immediate application is for explainable planning [Fox *et al.* 2017; Seegebarth *et al.* 2012]. Indeed a high-level plan is more concise and does not contain unnecessary implementation details that would confuse a non-expert. Recent works focus on matching the domain to a separate human model [Sreedharan and Chakraborti *et al.* 2018; Sreedharan and Srivastava *et al.* 2018]. This requires the creation and maintenance of expansive human domains that are mostly used as dictionaries to explain technical details. Our method will give coherent high level plans that are more concise and simpler than any classical plans.

Another potential application for such plans is relative to domains that work with approximative data. Our main example here is intent recognition which is the original motivation for this work. Planners are not meant to solve intent recognition problems. However, several works extended what is called in psychology the *theory of mind*. That theory is the equivalent of asking "*what would **I** do if I was them ?*" when observing the behavior of other agents. This leads to new ways to use *inverted planning* as an inference tool. One of the first to propose that idea was Baker *et al.* [2007] that use Bayesian planning to infer intentions. Ramirez and Geffner [2009] found an elegant way to transform a plan recognition problem into classical planning. This is done simply by encoding temporal constraints in the planning domain in a similar way as Baioletti *et al.* [1998] describe it to match the observed action sequence. A cost comparison will then give a probability of the goal to be pursued given the observations. A new method, proposed by Sohrabi *et al.* [2016], makes the recognition fluent centric. It assigns costs to missing or noisy observed fluents, which allows finer details and less preprocessing work than action-based recognition. Sohrabi *et al.* state that the quality of the recognition is directly linked to the quality and domain coverage of the generated plans.

2 Related Works

HTN is often combined with classical approaches since it allows for a more natural expression of domains making expert knowledge easier to encode. These kinds of planners are named **decompositional planners** when no initial plan is provided [Fox 1997]. Most of the time the integration of HTN simply consists in calling another algorithm when introducing a composite operator during the planning process. In the case of the DUET planner by Gerevini *et al.* [2008], it is done by calling an instance of an HTN planner based on task insertion called SHOP2 [Nau *et al.* 2003] to deal with composite actions. Some planners take the integration further by making the decomposition of composite actions into a special step in their refinement process. Such works include the discourse generation oriented DPOCL [Young and Moore 1994] and the work of Kambhampati *et al.* [1998] generalizing the practice for decompositional planners.

In our case, we chose a class of hierarchical planners based on Plan Space Planning (PSP) algorithms [Bechon *et al.* 2014; Dvorak *et al.* 2014; Bercher *et al.* 2014] as a reference approach. The main difference here is that the decomposition is integrated into the classical POCL algorithm by only adding new types of flaws. This allows to keep all the flexibility and properties of POCL while adding the expressivity and abstraction capabilities of HTN. We also designed an improved planning framework based on the one used by HiPOP to reduce further the number of changes needed to handle composite actions and to increase the planner efficiency.

Another work has already been done on another aspect of those types of abstract plans. The Angelic algorithm by Marthi *et al.* [2007] exploited the usefulness of such plans in the planning process itself and used them as a heuristic guide. They also proved that, for a given fluent semantics, it is guaranteed that such abstract solutions can be refined into actual solutions. However, the Angelic planner does not address the inherent properties of such abstract plans as approximate solutions and uses a more restrictive totally ordered framework.

3 Definitions

3.1 Domain

The domain specifies the allowed operators that can be used to plan and all the fluents they use as preconditions and effects.

Definition 1 (Domain)**.** A domain is a triplet $\mathcal{D} = \langle E_\mathcal{D}, R, A_\mathcal{D} \rangle$ where:

- $E_\mathcal{D}$ is the set of **domain entities**.
- R is the set of **relations** over $E_\mathcal{D}^n$. These relations are akin to n-ary predicates in first order logic.
- $A_\mathcal{D}$ is the set of **operators** which are fully lifted *actions*.

Fluents are signed first order logic n-ary predicates. Negative fluents are noted $\neg f$ and behave as a logical complement. We do not use the closed world hypothesis: fluents are only satisfied when another compatible fluent is provided.

Example: To describe an item not being held, we use the fluent $\neg taken(item)$. If the cup contains water, $in(water, cup)$ is true.

3.2 Plan and hierarchical representation

Definition 2 (Partial Plan / Method)**.** A partially ordered plan is an *acyclic* directed graph $\pi = (S, L)$, with:

- S the set of **steps** of the plan as vertices. A step is an action belonging in the plan. S must contain an initial step I_π and goal step G_π.
- L the set of **causal links** of the plan as edges. We note $l = a_s \xrightarrow{c} a_t$ the link between its source a_s and its target a_t caused by the set of fluents c. If $c = \varnothing$ then the link is used as an ordering constraint.

In our framework, *ordering constraints* are defined as the transitive cover of causal links over the set of steps. We note ordering constraints: $a_a \succ a_s$, with a_a being *anterior* to its *successor* a_s. Ordering constraints cannot form cycles, meaning that the steps must be different and that the successor cannot also be anterior to its anterior steps: $a_a \neq a_s \wedge a_s \not\succ a_a$. In all plans, the initial and goal steps have their order guaranteed:

$I_\pi \succ G\pi \wedge \nexists a_x \in S_\pi : a_x \succ I_\pi \vee G_\pi \succ a_x$. If we need to enforce order, we simply add a link without specifying a cause. The use of graphs and implicit order constraints help to simplify the model while maintaining its properties.

The central notion of planning is operators. Instantiated operators are usually called *actions*. In our framework, actions can be partially instantiated. We use the term action for both lifted and grounded operators.

Definition 3 (Action)**.** An action is a parametrized tuple $a(args) = \langle name, pre, eff, methods \rangle$ where:

- $name$ is the **name** of the action.
- pre and eff are sets of fluents that are respectively the **preconditions and the effects** of the action.
- $methods$ is a set of **methods** (*partial order plans*) that decompose the action into smaller ones. Methods, and the methods of their enclosed actions, cannot contain the parent action.

Example: The precondition of the operator $take(item)$ is simply a single negative fluent noted $\neg taken(item)$ ensuring the variable $item$ is not already taken.

Composite actions are represented using methods. An action without methods is called *atomic*. It is of interest to note the divergence with classical HTN representation here since normally composite actions do not have preconditions nor effects. In our case we insert them into abstract plans.

In order to verify the input of the domain, the causes of the causal links in the methods are optional. If omitted, the causes are inferred by unifying the preconditions and effects with the same mechanism as in the subgoal resolution in our POCL algorithm. Since we want to guarantee the validity of abstract plans, we need to ensure that user provided plans are solvable. We use the following formula to compute the final preconditions and effects of any composite action a: $pre(a) = \bigcup_{a_s \in L^+(a)} causes(a_s)$ and $eff(a) = \bigcup_{a_s \in L^-(a)} causes(a_s)$. An instance of the classical POCL algorithm is then run on the problem $\mathcal{P}_a = \langle \mathcal{D}, C_\mathcal{P}, a \rangle$ to ensure its coherence. The domain compilation fails if POCL cannot be completed. Since our decomposition hierarchy is acyclic ($a \notin A_a$, see definition 9) nested methods cannot contain their parent action as a step.

3.3 Problem

Problem instances are often most simply described by two components: the initial state and the goal.

Definition 4 (Problem)**.** The planning problem is defined as a tuple $\mathcal{P} = \langle \mathcal{D}, C_\mathcal{P}, a_0 \rangle$ where:

- $\mathcal{D}$ is a planning domain.
- $C_\mathcal{P}$ is the set of **problem constants** disjoint from the domain constants.
- a_0 is the **root operator** of the problem which methods are potential solutions of the problem.

Example: We use a simple problem for our example domain. The initial state provides that nothing is ready, taken or hot and all containers are empty (all using quantifiers). The goal is to have tea made.

The root operator is initialized to $a_0 = \langle \text{""}, s_0, s^*, \{\pi_{lv(a_0)}\} \rangle$, with s_0 being the initial state and s^* the goal specification. The method $\pi_{lv(a_0)}$ is a partial order plan with the initial and goal steps linked together via a_0. The initial partial order plan is $\pi_{lv(a_0)} = (\{I, G\}, \{I \xrightarrow{s_0} a_0 \xrightarrow{s^*} G\})$, with $I = \langle \text{"}init\text{"}, \emptyset, s_0, \emptyset \rangle$ and $G = \langle \text{"}goal\text{"}, s^*, \emptyset, \emptyset \rangle$.

3.4 Partial Order Causal Links

Our method is based on the classical POCL algorithm. It works by refining a partial plan into a solution by recursively removing all of its flaws.

Definition 5 (Flaws)**.** Flaws have a *proper fluent* f and a causing step often called the *needer* a_n. Flaws in a partial plan are either:

- **Subgoals**, *open conditions* that are yet to be supported by another step a_n often called *provider*. We note subgoals $\ddagger_f a_n$.
- **Threats**, caused by steps that can break a causal link with their effects. They are called *breakers* of the threatened link. A step a_b threatens a causal link $l_t = a_p \xrightarrow{f} a_n$ if and only if $\neg f \in eff(a_b) \wedge a_b \not\succ a_p \wedge a_n \not\succ a_b$. Said otherwise, the breaker can cancel an effect of a providing step a_p, before it gets used by its needer a_n. We note threats $a_b \otimes l_t$.

Example: Our initial plan contains two unsupported subgoals: one to make the tea ready and another to put sugar in it. In this case, the needer is the goal step and the proper fluents are each of its preconditions.

These flaws need to be fixed in order for the plan to be valid. In POCL it is done by finding their resolvers.

Definition 6 (Resolvers)**.** Classical resolvers are additional causal links that aim to fix a flaw.

- *For subgoals*, the resolvers are a set of potential causal links containing the proper fluent f in their causes while taking the needer step a_n as their target and a **provider** step a_p as their source.
- *For threats*, we usually consider only two resolvers: **demotion** ($a_b \succ a_p$) and **promotion** ($a_n \succ a_b$) of the breaker relative to the threatened link. We call the added causeless causal link a **guarding** link.

Example: The subgoal for $in(water, cup)$, in our example, can be solved by using the action $pour(water, cup)$ as the source of a causal link carrying the proper fluent as its only cause.

The application of a resolver does not necessarily mean progress. It can have consequences that may require reverting its application in order to respect the backtracking of the POCL algorithm.

Definition 7 (Side effects)**.** Flaws that are caused by the application of a resolver are called *related flaws*. They are inserted into the *agenda*[1] with each application of a resolver:

- *Related subgoals* are all the new open conditions inserted by new steps.
- *Related threats* are the causal links threatened by the insertion of a new step or the deletion of a guarding link.

Flaws can also become irrelevant when a resolver is applied. Those *invalidated flaws* are removed from the agenda upon detection:

[1] An agenda is a flaw container used for the flaw selection of POCL.

- *Invalidated subgoals* are subgoals satisfied by the new causal links or the removal of their needer.
- *Invalidated threats* happen when the breaker no longer threatens the causal link because the order guards the threatened causal link or either of them have been removed.

Example: Adding the action $pour(water, cup)$ causes a related subgoal for each of the preconditions of the action which are: the cup and the water must be taken and water must not already be in the cup.

In algorithm 1 we present a generic version of POCL inspired by Ghallab *et al.* [2004, sec. 5.4.2].

Algorithm 1 Partial Order Planner

```
1  function POCL(Agenda a, Problem P)
2      if a = ∅ then                          ▷ Populated agenda needs to be provided
3          return Success                     ▷ Stops all recursion
4      Flaw f ← choose(a)                     ▷ Heuristically chosen flaw
5      Resolvers R ← solve(f, P)
6      for all r ∈ R do                       ▷ Non-deterministic choice operator
7          apply(r, π)                        ▷ Apply resolver to partial plan
8          Agenda a' ← update(a)
9          if POCL(a', P) = Success then      ▷ Refining recursively
10             return Success
11         revert(r, π)                       ▷ Failure, undo resolver application
12     a ← a ∪ {f}                            ▷ Flaw was not resolved
13     return Failure                         ▷ Revert to last non-deterministic choice
```

Resolvers are picked non-deterministically for applications (this can be heuristically driven). At line 7 the resolver is effectively applied to the current plan. All side effects and invalidations are handled during the update of the agenda at line 8. If a problem occurs, line 11 backtracks and tries other resolvers. If no resolver fits the flaw, the algorithm backtracks to previous resolver choices to explore all the possible plans and ensure completeness.

In definition 7, we mentioned effects that aren't present in classical POCL, namely *negative resolvers*. All classical resolvers only add steps and causal links to the partial plan. Our method needs to remove composite steps and their adjacent links when expanding them.

4 The Heart of the Method

In this section, we explain how our method combines POCL with HTN planning and how they are used to generate intermediary abstract plans.

4.1 Additional Notions

In order to properly introduce the changes made for using HTN domains in POCL, we need to define a few notions.

Transposition is needed to define decomposition.

Definition 8 (Transposition)**.** In order to transpose the causal links of an action a' with the ones of an existing step a in a plan π, we use the following operation :

$$a \rhd^-_\pi a' = \left\{ \phi^-(l) \xrightarrow{causes(l)} a' \ : \ l \in L^-_\pi(a) \right\}$$

It is the same with $a' \xrightarrow{causes(l)} \phi^+(l)$ and L^+ for $a \rhd^+ a'$. This supposes that the respective preconditions and effects of a and a' are equivalent. When not signed, the transposition is generalized: $a \rhd a' = a \rhd^- a' \cup a \rhd^+ a'$.

Example: $a \rhd^- a'$ gives all incoming links of a with the target set to a' instead.

Definition 9 (Proper Actions)**.** Proper actions are actions that are "contained" within an entity. We note this notion $A_a = A_a^{lv(a)}$ for an action a. It can be applied to various concepts :

- For a *domain* or a *problem*, $A_\mathcal{P} = A_\mathcal{D}$.
- For a *plan*, it is $A^0_\pi = S_\pi$.
- For an *action*, it is $A^0_a = \bigcup_{m \in methods(a)} S_m$. Recursively: $A^n_a = \bigcup_{b \in A^0_a} A^{n-1}_b$. For atomic actions, $A_a = \varnothing$.

Example: The proper actions of $make(drink)$ are the actions contained within its methods. The set of extended proper actions adds all proper actions of its single proper composite action $infuse(drink, water, cup)$.

Definition 10 (Abstraction Level)**.** This is a measure of the maximum amount of abstraction an entity can express:[2]

$$lv(x) = \left(\max_{a \in A_x} (lv(a)) + 1 \right) [A_x \neq \varnothing]$$

Example: The abstraction level of any atomic action is 0 while it is 2 for the composite action $make(drink)$. The example domain has an abstraction level of 3.

4.2 Abstraction In POCL

The most straightforward way to handle abstraction in regular planners is illustrated by Duet [Gerevini *et al.* 2008] by managing hierarchical actions separately from a task insertion planner. We chose to add abstraction in POCL in a manner inspired by the work of Bechon *et al.* [2014] on a planner called HiPOP. The difference between the original HiPOP and our implementation of it is that we focus on the expressivity and the ways flaw selection can be exploited for partial resolution. Our version is lifted at runtime while the original is grounded for optimizations. All mechanisms we have implemented use POCL but with different management of flaws and resolvers. The original algorithm 1 is left untouched.

One of those changes is that resolver selection needs to be altered for subgoals. Indeed, as stated by the authors of HiPOP : the planner must ensure the selection of high-level operators in order to benefit from the hierarchical aspect of the domain, otherwise, adding operators only increases the branching factor. We also need to add a way to deal with composite actions once inserted in the plan to reduce them to their atomic steps.

Definition 11 (Decomposition Flaws)**.** They occur when a partial plan contains a non-atomic step. This step is the needer a_n of the flaw. We note its decomposition $a_n\oplus$.

- *Resolvers*: A decomposition flaw is solved with a **decomposition resolver**. The resolver will replace the needer with one of its instantiated methods $m \in methods(a_n)$ in the plan π. This is done by using transposition such that: $a_n \oplus^m_\pi = \langle S_m \cup (S_\pi \setminus \{a\}), a_n \rhd^- I_m \cup a_n \rhd^+ G_m \cup (L_\pi \setminus L_\pi(a_n)) \rangle$.

[2] We use Iverson brackets here.

- *Side effects*: A decomposition flaw can be created by the insertion of a composite action in the plan by any resolver and invalidated by its removal :

$$\bigcup_{a_m \in S_m}^{f \in pre(a_m)} \pi' \ddagger_f a_m \bigcup_{a_b \in S_{\pi'}}^{l \in L_{\pi'}} a_b \otimes l \bigcup_{a_c \in S_m}^{lv(a_c) \neq 0} a_c \oplus$$

Example: When adding the step $make(tea)$ in the plan to solve the subgoal that needs tea being made, we also introduce a decomposition flaw that will need this composite step replaced by its method using a decomposition resolver. In order to decompose a composite action into a plan, all existing links are transposed to the initial and goal step of the selected method, while the composite action and its links are removed from the plan. The main differences between HiPOP and HEART in our implementations are the functions of flaw selection and the handling of the results (one plan for HiPOP and a plan per cycle for HEART). In HiPOP, the flaw selection is made by prioritizing the decomposition flaws. Bechon *et al.* [2014] state that it makes the full resolution faster. However, it also loses opportunities to obtain abstract plans in the process.

4.3 Cycles

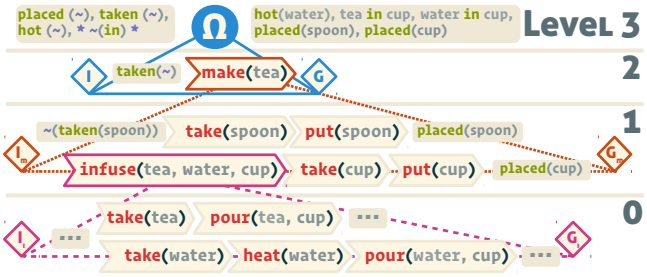


Figure 1: Illustration of how the cyclical approach is applied on the example domain. Atomic actions that are copied from a cycle to the next are omitted.

The main focus of our work is toward obtaining **abstract plans** which are plans that are completed while still containing composite actions. In order to do that the flaw selection function will enforce cycles in the planning process.

Definition 12 (Cycle)**.** A cycle is a planning phase defined as a triplet $c = \langle lv(c), agenda, \pi_{lv(c)} \rangle$ where : $lv(c)$ is the maximum abstraction level allowed for flaw selection in the $agenda$ of remaining flaws in partial plan $\pi_{lv(c)}$. The resolvers of subgoals are therefore constrained by the following: $a_p \downarrow_f a_n : lv(a_p) \leq lv(c)$.

During a cycle all decomposition flaws are delayed. Once no more flaws other than decomposition flaws are present in the agenda, the current plan is saved and all remaining decomposition flaws are solved at once before the abstraction level is lowered for the next cycle: $lv(c') = lv(c) - 1$. Each cycle produces a more detailed abstract plan than the one before.

Abstract plans allow the planner to do an approximate form of anytime execution. At any given time the planner is able to return a fully supported plan. Before the first cycle, the plan returned is $\pi_{lv(a_0)}$.

Example: In our case using the method of intent recognition of Sohrabi *et al.* [2016], we can already use $\pi_{lv(a_0)}$ to find a likely goal explaining an observation (a set of temporally ordered fluents). That can make an early assessment of the probability of each goal of the recognition problem.

For each cycle c, a new plan $\pi_{lv(c)}$ is created as a new method of the root operator a_0. These intermediary plans are not solutions of the problem, nor do they mean that the problem is solvable. In order to find a solution, the HEART planner needs to reach the final cycle c_0 with an abstraction level $lv(c_0) = 0$. However, these plans can be used to derive meaning from the potential solution of the current problem and give a good approximation of the final result before its completion.

Example: In the figure 1, we illustrate the way our problem instance is progressively solved. Before the first cycle c_2, all we have is the root operator and its plan π_3. Then within the first cycle, we select the composite action $make(tea)$ instantiated from the operator $make(drink)$ along with its methods. All related flaws are fixed until all that is left in the agenda is the abstract flaws. We save the partial plan π_2 for this cycle and expand $make(tea)$ into a copy of the current plan π_1 for the next cycle. The solution of the problem will be stored in π_0 once found.

5 Theoretical analysis

In this section, we prove several properties of our method and resulting plans : HEART is complete, sound and its abstract plans can always be decomposed into a valid solution.

The completeness and soundness of POCL has been proven in [Penberthy *et al.* 1992]. An interesting property of POCL algorithms is that flaw selection strategies do not impact these properties. Since the only modification of the algorithm is the extension of the classical flaws with a decomposition flaw, all we need to explore, to update the proofs, is the impact of the new resolver. By definition, the resolvers of decomposition flaws will take into account all flaws introduced by its resolution into the refined plan. It can also revert its application properly.

Lemma (Decomposing preserves acyclicity)**.** *The decomposition of a composite action with a valid method in an acyclic plan will result in an acyclic plan. Formely,* $\forall a_s \in S_\pi : a_s \not\succ_\pi a_s \implies \forall a'_s \in S_{a \oplus^m_\pi} : a'_s \not\succ_{a \oplus^m_\pi} a'_s$.

Proof. When decomposing a composite action a with a method m in an existing plan π, we add all steps S_m in the refined plan. Both π and m are guaranteed to be cycle free by definition. We can note that $\forall a_s \in S_m : (\nexists a_t \in S_m : a_s \succ a_t \wedge \neg f \in \mathit{eff}(a_t)) \implies f \in \mathit{eff}(a)$. Said otherwise, if an action a_s can participate a fluent f to the goal step of the method m then it is necessarily present in the effects of a. Since higher level actions are preferred during the resolver selection, no actions in the methods are already used in the plan when the decomposition happens. This can be noted $\exists a \in \pi \implies S_m \cup S_\pi$ meaning that in the graph formed both partial plans m and π cannot contain the same edges therefore their acyclicity is preserved when inserting one into the other. □

Lemma (Solved decomposition flaws cannot reoccur)**.** *The application of a decomposition resolver on a plan* π*, guarantees that* $a \notin S_{\pi'}$ *for any partial plan refined from* π *without reverting the application of the resolver.*

Proof. As stated in the definition of the methods (definition 3): $a \notin A_a$. This means that a cannot be introduced in the plan by its decomposition or the decomposition of its proper actions. Indeed, once a is expanded, the level of the following cycle $c_{lv(a)-1}$ prevents a to be selected by subgoal resolvers. It cannot either be contained in the methods of another action that are selected afterward because otherwise following definition 10 its level would be at least $lv(a) + 1$.

□

Lemma (Decomposing to abstraction level 0 guarantees solvability)**.** *Finding a partial plan that contains only decomposition flaws with actions of abstraction level 1, guarantees a solution to the problem.*

Proof. Any method m of a composite action $a : lv(a) = 1$ is by definition a solution of the problem $\mathcal{P}_a = \langle \mathcal{D}, C_\mathcal{P}, a \rangle$. By definition, $a \notin A_a$, and $a \notin A_{a \oplus_\pi^m}$ (meaning that a cannot reoccur after being decomposed). It is also given by definition that the instantiation of the action and its methods are coherent regarding variable constraints (everything is instantiated before selection by the resolvers). Since the plan π only has decomposition flaws and all flaws within m are guaranteed to be solvable, and both are guaranteed to be acyclical by the application of any decomposition $a \oplus_\pi^m$, the plan is solvable.

□

Lemma (Abstract plans guarantee solvability)**.** *Finding a partial plan π that contains only decomposition flaws, guarantees a solution to the problem.*

Proof. Recursively, if we apply the previous proof on higher level plans we note that decomposing at level 2 guarantees a solution since the method of the composite actions are guaranteed to be solvable.

□

From these proofs, we can derive the property of soundness (from the guarantee that the composite action provides its effects from any methods) and completeness (since if a composite action cannot be used, the planner defaults to using any action of the domain).

6 Experimental evaluation

In order to assess its capabilities, HEART was evaluated on two criteria: quality and complexity. All tests were executed on an Intel® Core™ i7-7700HQ CPU clocked at 2.80GHz. The Java process used only one core and was not limited by time or memory.[3] Each experiment was repeated between 700 and 10 000 times to ensure that variations in speed were not impacting the results.

Figure 2 shows how the quality is affected by the abstraction in partial plans. The tests are made using our example domain. The quality is measured by counting the number of providing fluents in the plan $\left|\bigcup_{a \in S_\pi} eff(a)\right|$. This metric is actually used to approximate the probability of a goal given observations in intent recognition ($P(G|O)$ with noisy observations, see [Sohrabi *et al.* 2016]). The percentages are relative to the total number

[3]The source code of HEART will be available at genn.io/heart

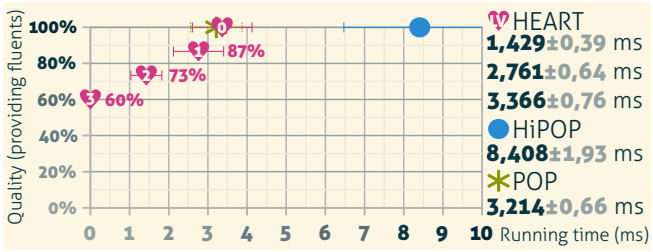


Figure 2: Evolution of the quality with computation time.

of unique fluents of the complete solution. These results show that in some cases it may be more interesting to plan in a leveled fashion to solve HTN problems. For the first cycle of level 3, the quality of the abstract plan is already of 60%. This is the quality of the exploitation of the plan *before any planning*. With almost three quarters of the final quality and less than half of the complete computation time, the result of the first cycle is a good quality/time compromise.

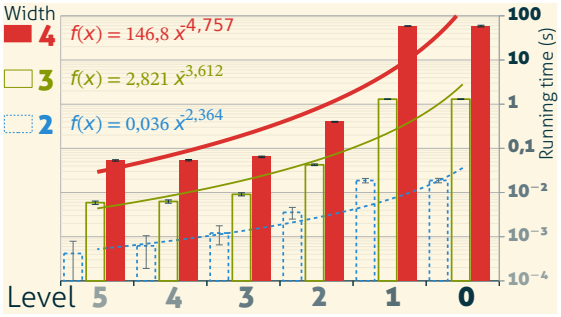


Figure 3: Impact of domain shape on the computation time by levels. The scale of the vertical axis is logarithmic. Equations are the definition of the trend curves.

In the second test, we used generated domains. These domains consist of an action of abstraction level 5. This action has a single method containing a number of actions of level 4. We call this number the width of the domain. All needed actions are built recursively to form a tree shape. Atomic actions only have single fluent effects. The goal is the effect of the higher level action and the initial state is empty. These domains do not contain negative effects. Figure 3 shows the computational profile of HEART for various levels and widths. We note that the behavior of HEART seems to follow an exponential law with the negative exponent of the trend curves seemingly being correlated to the actual width. This means that computing the first cycles has a complexity that is close to being *linear* while computing the last cycles is of the same complexity as classical planning which is at least *P-SPACE* (depending on the expressivity of the domain) [Erol *et al.* 1995].

Conclusion

In this paper, we have presented a new planner called HEART based on POCL. An updated planning framework fitting the need for such a new approach was proposed. We showed how HEART performs compared to complete planners in terms of speed and quality. While the abstract plans generated during the planning

process are not complete solutions, they are exponentially faster to generate while retaining significant quality over the final plans. They are also proof of solvability of the problem. By using these plans, it is possible to generate explanations of intractable problems within tight time constraints.

References

[Baioletti *et al.* 1998] Baioletti Marco, Stefano Marcugini, and Alfredo Milani Encoding planning constraints into partial order planning domains. *International Conference on Principles of Knowledge Representation and Reasoning*, 608–616 Morgan Kaufmann Publishers Inc., 1998.

[Baker *et al.* 2007] Baker Chris L., Joshua B. Tenenbaum, and Rebecca R. Saxe Goal inference as inverse planning. *Proceedings of the Annual Meeting of the Cognitive Science Society*, Vol. 29, 2007.

[Bechon *et al.* 2014] Bechon Patrick, Magali Barbier, Guillaume Infantes, Charles Lesire, and Vincent Vidal HiPOP: Hierarchical Partial-Order Planning. *European Starting AI Researcher Symposium*, 264,51–60 IOS Press, 2014.

[Bercher *et al.* 2014] Bercher Pascal, Shawn Keen, and Susanne Biundo Hybrid planning heuristics based on task decomposition graphs. *Seventh Annual Symposium on Combinatorial Search*, 2014.

[Biundo and Schattenberg 2001] Biundo S., and B. Schattenberg From abstract crisis to concrete relief preliminary report on flexible integration on nonlinear and hierarchical planning. *Proceedings of the European Conference on Planning*, 2001.

[Dvorak *et al.* 2014] Dvorak Filip, Arthur Bit-Monnot, Félix Ingrand, and Malik Ghallab A flexible ANML actor and planner in robotics. *Planning and Robotics (PlanRob) Workshop (ICAPS)*, 2014.

[Erol *et al.* 1994] Erol Kutluhan, James Hendler, and Dana S. Nau HTN planning: Complexity and expressivity. *AAAI*, 94,1123–1128 1994.

[Erol *et al.* 1995] Erol Kutluhan, Dana S. Nau, and Venkatramana S. Subrahmanian Complexity, decidability and undecidability results for domain-independent planning. *Artificial intelligence*, 76 (1-2), 75–88, 1995.

[Fox 1997] Fox Maria Natural hierarchical planning using operator decomposition. *European Conference on Planning*, 195–207 Springer, 1997.

[Fox *et al.* 2017] Fox Maria, Derek Long, and Daniele Magazzeni Explainable Planning. *Proceedings of IJCAI Workshop on Explainable AI*, Melbourne, Australia, August 2017.

[Gerevini *et al.* 2008] Gerevini Alfonso, Ugur Kuter, Dana S. Nau, Alessandro Saetti, and Nathaniel Waisbrot Combining Domain-Independent Planning and HTN Planning: The Duet Planner. *Proceedings of the European Conference on Artificial Intelligence*, 18,573–577 2008.

[Ghallab *et al.* 2004] Ghallab Malik, Dana Nau, and Paolo Traverso *Automated planning: Theory & practice*, Elsevier, 2004.

[Ghallab *et al.* 2016] Ghallab Malik, Dana Nau, and Paolo Traverso *Automated Planning and Acting*, Cambridge University Press, 2016.

[Göbelbecker *et al.* 2010] Göbelbecker Moritz, Thomas Keller, Patrick Eyerich, Michael Brenner, and Bernhard Nebel Coming Up With Good Excuses: What to do When no Plan Can be Found. *Proceedings of the International Conference on Automated Planning and Scheduling*, 20,81–88 AAAI Press, May 2010.

[Kambhampati *et al.* 1998] Kambhampati Subbarao, Amol Mali, and Biplav Srivastava Hybrid planning for partially hierarchical domains. *AAAI/IAAI*, 882–888 1998.

[Lemai and Ingrand 2004] Lemai Solange, and Félix Ingrand Interleaving temporal planning and execution in robotics domains. *AAAI*, 4,617–622 2004.

[Marthi *et al.* 2007] Marthi Bhaskara, Stuart J. Russell, and Jason Andrew Wolfe Angelic Semantics for High-Level Actions. *ICAPS*, 232–239 2007.

[Nau *et al.* 2003] Nau Dana S., Tsz-Chiu Au, Okhtay Ilghami, Ugur Kuter, J. William Murdock, et al. SHOP2: An HTN planning system. *J. Artif. Intell. Res.(JAIR)*, 20, 379–404, 2003.

[Penberthy *et al.* 1992] Penberthy J Scott, Daniel S Weld, and others UCPOP: A Sound, Complete, Partial Order Planner for ADL. *Kr*, 92, 103–114, 1992.

[Ramirez and Geffner 2009] Ramirez Miquel, and Hector Geffner Plan recognition as planning. *Proceedings of the International Conference on International Conference on Automated Planning and Scheduling*, 19,1778–1783 AAAI Press, 2009.

[Sacerdoti 1974] Sacerdoti Earl D. Planning in a hierarchy of abstraction spaces. *Artificial intelligence*, 5 (2), 115–135, 1974.

[Sacerdoti 1975] Sacerdoti Earl D. *The nonlinear nature of plans*, STANFORD RESEARCH INST MENLO PARK CA, 1975.

[Seegebarth *et al.* 2012] Seegebarth Bastian, Felix Müller, Bernd Schattenberg, and Susanne Biundo Making hybrid plans more clear to human usersa formal approach for generating sound explanations. *Proceedings of the Twenty-Second International Conference on International Conference on Automated Planning and Scheduling*, 225–233 AAAI Press, 2012.

[Sohrabi *et al.* 2016] Sohrabi Shirin, Anton V. Riabov, and Octavian Udrea Plan Recognition as Planning Revisited. *Proceedings of the International Joint Conference on Artificial Intelligence*, Vol. 25, 2016.

[Sreedharan *et al.* 2018] Sreedharan Sarath, Tathagata Chakraborti, and Subbarao Kambhampati Handling Model Uncertainty and Multiplicity in Explanations via Model Reconciliation. *International Conference on Automated Planning and Scheduling*, 2018.

[Sreedharan *et al.* 2018] Sreedharan Sarath, Siddharth Srivastava, and Subbarao Kambhampati Hierarchical Expertise-Level Modeling for User Specific Robot-Behavior Explanations. *International Joint Conference on Artificial Intelligence*, 2018.

[Weld 1994] Weld Daniel S. An introduction to least commitment planning. *AI magazine*, 15 (4), 27, 1994.

[Young and Moore 1994] Young R. Michael, and Johanna D. Moore DPOCL: A principled approach to discourse planning. *Proceedings of the Seventh International Workshop on Natural Language Generation*, 13–20 Association for Computational Linguistics, 1994.

A Qualitative Analysis of Search Behavior: A Visual Approach

Ian Howell,[1,2] **Robert Woodward,**[1] **Berthe Y. Choueiry,**[1] **Hongfeng Yu**[2]
[1]Constraint Systems Laboratory, University of Nebraska-Lincoln, USA
[2]Visualization Laboratory, University of Nebraska-Lincoln, USA
{ihowell|rwoodwar|choueiry|yu}@cse.unl.edu

Abstract

In this paper, we propose visualizations that track the progress and behavior of backtrack search when solving an instance of a Constraint Satisfaction Problem. The goal of our visualizations is to provide insight in the difficulty of the particular instance at hand as well as in the effectiveness of various strategies for enforcing consistency during search. To this end, our visualizations track the number of backtracks and the number of calls to a consistency algorithm per depth of the search tree and superimpose the two measures while distinguishing effective and wasteful consistency calls. Using these numbers, we automatically derive qualitative regimes summarizing the evolution of the search process over time. We show that these instruments provide new insights into the performance of search on a particular instance and into the effectiveness of the various strategies for enforcing consistency during search. We present WORMHOLE, an extendable, solver-agnostic visualization tool that we built as a platform to implement these mechanisms. Currently WORMHOLE provides a 'post-mortem' analysis of search, but our ultimate goal is to provide an 'in-vivo' analysis and allow the user to intervene and guide the search process.

1 Introduction

In this paper, we propose a new perspective and visualization tools to understand and analyze the behavior of the backtrack-search procedure for solving Constraint Satisfaction Problems (CSPs). Backtrack search is currently the only sound and complete algorithm for solving CSPs. However, its performance is unpredictable and can differ widely on similar instances. Further, maintaining a given consistency property during search has become a common practice and new strategies for dynamically switching between consistency algorithms are being investigated [Borrett *et al.*, 1996; Freuder and Wallace, 1991; Epstein *et al.*, 2005; Eén and Biere, 2005; Stergiou, 2008; Lagerkvist and Schulte, 2009; Paparrizou and Stergiou, 2012; Balafrej *et al.*, 2013; Balafrej *et al.*, 2014; Woodward *et al.*, 2014; Wallace, 2015; Balafrej *et al.*, 2015; Woodward *et al.*, 2011; Paparrizou and Stergiou, 2017]. While consistency algorithms can dramatically reduce the size of the search space, their impact on the CPU cost of search can vary widely. It is likely poorly understood but certainly difficult to control.

In this paper, we propose three tools and their visualizations as a first step towards graphically summarizing and explaining the performance of search:

1. We track the number of *backtracks per depth* (BpD) at each level of search to understand where and how search struggles and where it smoothly proceeds.
2. To understand the impact of enforcing a given consistency property, we track the number of *calls to the consistency algorithm per depth* (CpD) in the search tree. Further, we split these calls into three categories: those that yield domain *wipeout* (i.e., detect inconsistency), those that effectively *filter* domains without detecting a dead-end, and those that yield *no filtering* (i.e., constitute a wasted effort).
3. To summarize the behavior of search over time, we structure the evolution of the above two measurements into qualitatively equivalent *regimes* by using three criteria that characterize growth rate and shape evolution.

We implement the above mechanisms in a visualization system called WORMHOLE that allows users to interactively examine the performance of search and graphically compare the performance of various algorithms on the same instance. WORMHOLE is a first step towards building a library of visualization tools aimed at providing an insight into the strengths and weaknesses of current algorithms for solving CSPs.

While our system does not generate verbal explanations, we claim that the graphical tools that it provides directly 'speak' to a user's intuitions. Our long-term goal is to allow users to actively intervene in the search process itself, trying alternatives and mixing strategies while observing their effects on the effectiveness of problem solving.

This paper is structured as follows. Section 2 recalls background information and reviews the relevant literature. Section 3 describes the proposed visualizations. Section 4 discusses how search evolution is organized into regimes of equivalent behaviors to facilitate user understanding. Section 5 is a case study showing how our visualizations allow

us to compare and understand the performance of three strategies for solving the same CSP instance. Section 6 briefly describes the architecture of WORMHOLE. Finally, Section 7 concludes the paper.

2 Background

A CSP is defined by a tuple (X, D, C), where X is a set of variables, D is the set of the variables' domains, and C a set of constraints that restrict the combinations of values that the variables can take at the same time. A solution to a CSP is an assignment of values to variables such that all constraints are simultaneously satisfied.

Backtrack (BT) search is currently the only sound and complete algorithm for solving CSPs. In order to reduce thrashing, which is the main malady of search, it is common practice to enforce a given consistency property after every variable instantiation. This procedure reduces the size of the search space by deleting, from the variables' domains, values that cannot appear in a consistent solution given the current search path (i.e., conditioning). In recent years, the research community has investigated *higher-level consistencies* (HLC) as inference techniques to prune larger portions of the search space at the cost of increased processing effort [Freuder and Elfe, 1996; Debruyne and Bessiere, 1997; Samaras and Stergiou, 2005; Bessière *et al.*, 2008], leading to a tradeoff between the search effort and the time for enforcing consistency. We claim that our visualizations are insightful tools for understanding such a tradeoff.

Prior research on search visualization has appeared in the Constraint Programming literature, which typically focuses on the 2-way branching search scheme in contrast to the k-way branching scheme adopted by the CSP community. The DiSCiPl project provides extensive visual functionalities to develop, test, and debug constraint logic programs such as displaying variables' states, effect of constraints and global constraints, and event propagation at each node of the search tree [Simonis and Aggoun, 2000; Carro and Hermenegildo, 2000]. Many useful methodologies from the DiSCiPl project are implemented in CP-Viz [Simonis *et al.*, 2010] and other works [Shishmarev *et al.*, 2016]. The OZ Explorer displays the search tree allowing the user to access detailed information about the node at each tree node and to collapse and expand failing trees for closer examination [Schulte, 1996]. This work is currently incorporated into Gecode's Gist [Schulte *et al.*, 2015]. The above approaches focus on exploring the search tree (as well as a problem's components) while our work proposes particular projections (i.e., views, summaries) of the data *reflecting* (i.e., compiling) the cost and the effectiveness of both search and enforcing consistency. We believe that these visualizations are orthogonal and complementary.

Tracking search effort by depth was first proposed by Epstein *et al.* [2005] for the number of constraint checks and values removed per search and by Simonis *et al.* [2010] in CP-Viz for the number of nodes visited (also used for solving a packing problem [Simonis and O'Sullivan, 2011]). We claim that the number of constraint checks, values removed, and nodes visited are not accurate measures of the thrashing effort. Indeed, the number of constraint checks varies with the degree of the variables. The number of values removed and nodes visited vary with the size of the domain. In contrast, we claim that the number of backtracks per search depth (BpD) provides a more faithful representation of the thrashing effort, which is exactly the aspect of search that we aim to characterize.

Recently, techniques have appeared in Constraint Processing for dynamically choosing between a set of consistency properties based on the CPU time spent on exploring a given subtree [Balafrej *et al.*, 2015]. We claim that we better track the effectiveness of such decisions by following the number of backtracks per depth (BpD) and the number of consistency calls per depth (CpD) rather than the CPU time of searching a given subtree.

In the SAT community, inprocessing (in the form of the application of the resolution rule) interleaves search and inference steps [Järvisalo *et al.*, 2012; Wotzlaw *et al.*, 2013]. Resolution is allocated a fixed percentage of the CPU time (e.g., 10%) and sometimes its effectiveness is monitored for early termination. We believe that inference should be targeted at the 'areas' where search is struggling rather than following a predetermined and fixed effort allocation. We claim that the visualization provided by WORMHOLE can be used to identify where such an effort is best invested.

3 Analyzing Search Effectiveness

The BpD chart reflects various aspects of search effectiveness as we illustrate with an example. We consider the instance of a coloring problem called 4-insertions-3-3 of the benchmark graphColoring-k-insertions.[1] We find one solution of this instance using backtrack search while maintaining one of two consistency properties, namely GAC [Mackworth, 1977] and POAC [Bennaceur and Affane, 2001], and using the dom/wdeg ordering heuristic [Boussemart *et al.*, 2004]. The definitions of GAC and POAC are beyond the scope of this paper: it suffices to say that an algorithm that enforces GAC is generally quick but does little filtering while a POAC algorithm is typically (very) costly but can prune larger subtrees of the search space. In fact, enforcing POAC during search is so costly that, in practice, we use an adaptive version called APOAC [Balafrej *et al.*, 2014].

Table 1 reports the cost of the two search algorithms in terms of their CPU time, the number of nodes visited by search, the number of backtracks, and the maximum value reached by the respective BpD curves, which are shown in Figure 1. As we can see from Table 1, it is clear that our 'in-

Table 1: Cost of GAC and APOAC on 4-insertions-3-3. Note that GAC times out. In WORMHOLE, this data is displayed in a panel.

Algorithm	GAC	APOAC
CPU Time (sec)	>8,099.9	2,981.9
# Nodes Visited	348,276,252	17,078,644
# Backtracks	252,570,526	13,416,093
$\max_{\text{BpD}}$	15,863,603	693,829

[1] Source: `www.cril.univ-artois.fr/~lecoutre/benchmarks.html`.

vestment' in a stronger consistency algorithm is worthwhile because APOAC solves the instance in about 50 minutes while GAC does not terminate. Comparing the BpD charts of GAC (left) and APOAC (right) in Figure 1, we see that GAC thrashes around depth 50 with $\max_{\text{BpD}} = 15,863,603$ backtrack at depth 53. APOAC, which enforces a strictly stronger consistency throughout search, limits the severity of thrashing to only $\max_{\text{BpD}} = 693,829$ backtracks at depth 40. By detecting and pruning inconsistencies at the shallower search levels, APOAC solves the problem while GAC fails.

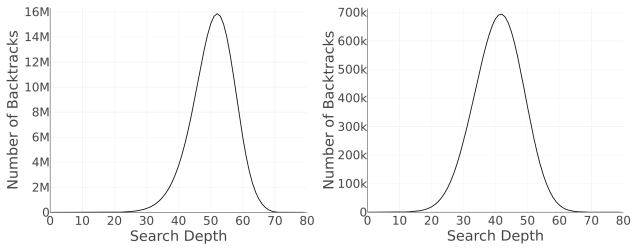


Figure 1: BpD for GAC (left) and APOAC (right) on instance 4-INSERTIONS-3-3.

Naturally, investing in a high-level consistency (HLC) is not always worthwhile. On easy instances, the cost of APOAC is can be an overkill. To examine the effectiveness of enforcing an HLC, we propose another visualization, which superimposes, to the BpD chart, the chart reporting the number of calls to POAC per depth (CpD). Figure 2 (left) unsurprisingly shows that the BpD and the CpD charts of APOAC largely overlap in shape (modulo their respective ranges shown on both sides of the chart), which is explained by the fact that APOAC is called at every variable instantiation during search. In other dynamic strategies where two or more levels of consistency are enforced, the CpD would allow us to differentiate between the impact of each consistency algorithm.

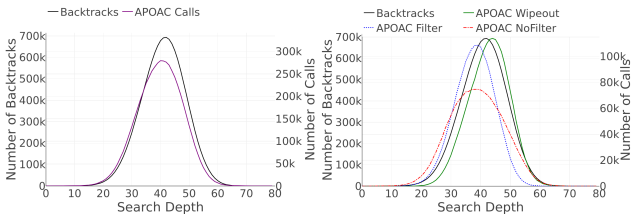


Figure 2: Superimposing CpD onto BpD for APOAC on 4-INSERTIONS-3-3: Showing CpD total (left) and detailed as wipeout, filtering, no filtering (right)

We propose a third visualization by splitting more finely the CpD into three categories depending on whether calls to POAC resulted in (1) in a domain wipeout, (2) filtering but no wipe out, and (3) no filtering. In Figure 2 (right), these three CpDs are shown in green (the most effective POAC calls, which cause backtracking), blue (which prune inconsistent subtrees, reduce the search space, but cannot detect inconsistencies), and red (which are wasteful calls to POAC resulting in no filtering). In the case of our particular example, we can see that the wasteful calls to POAC are fewer, and not as much time is wasted at deeper levels. This realization fully explains the ability of APOAC to prevent search from thrashing at deeper search levels and its effectiveness in solving this difficult instance.

4 Analyzing Search Evolution

The visualizations discussed in Section 3 provide interesting information at particular snapshots in time and at the end of search. However, in practice, search can last from a few milliseconds in duration to hours (before timeout). Thus, it is not practical, sometimes impossible, to examine the *evolution* of search from beginning to end. In order to help the user build an understanding of the evolution of search over time, gain insight in the difficulty of the problem at hand, perhaps even detect critical transitions in search behavior, we propose to automatically and dynamically organize the evolution of search over time in terms of a *history of successive episodes that are qualitatively meaningful*,[2] where each episode is a collection of equivalent behaviors.

We propose a two-step procedure to build such histories. First, we partition the time duration of the entire search into time intervals based on some criterion of equivalent behavior. We call these time intervals *regimes* [Kuipers, 1994]. Next, we provide a mechanism to dilate the time duration (i.e., by compressing or stretching it) of each regime, then concatenate them into a continuous animation of the history of the search.

Definition 1 (Regime, History) *Given a search procedure of total duration T, a given behavior function B and an equivalence relation $\sim$, a regime R_i is a contiguous interval of time $[s_i, e_i] \subseteq T$ during which the search features defining the behavior are qualitatively equivalent by some metric. The search history is a sequence of such regimes.*

$$
\begin{aligned}
H &= \langle R_1, \ldots, R_n \rangle \\
R_i &= \langle B_i, [s_i, e_i] \rangle \text{ where } [s_i, e_e] \subseteq T, \\
&\quad \forall t_i, t_j \in [s_i, e_e] \rightarrow B(t_i) \sim B_(t_j) \\
T &= \bigcup_{i=1}^{n} [s_i, e_i]
\end{aligned}
$$

While the set of desirable and useful behaviors and search features may be limitless, we integrate in WORMHOLE the following ones:

1. BASIC: Using the maximum value of BpD, $\max_{\text{BpD}}$, we partition the duration of search into a number of k regimes (k determined by the user), where the largest BpD value exceeds another k^{th} fraction of the value of $\max_{\text{BpD}}$.
2. GROWTH: From the beginning of search, we generate a new regime each time the largest BpD value increases by 10%.
3. SHAPE: From the start of search, we compute the Shannon Entropy of the derivatives over depth of the BpD to represent the relative shape of BpD curve [Rényi, 1961]. We start a new regime when this value changes by 20%.

[2] Our use of the term 'history' is in compliance with its initial meaning as proposed by Hayes [Hayes, 1990].

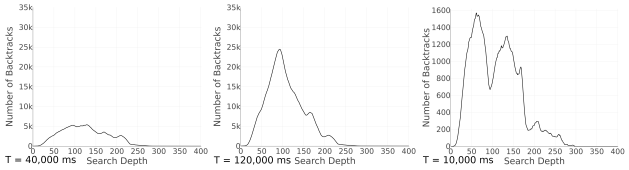


Figure 3: BASIC regimes of GAC on pseudo-aim-200-1-6-4 (cropped right-edges for space)

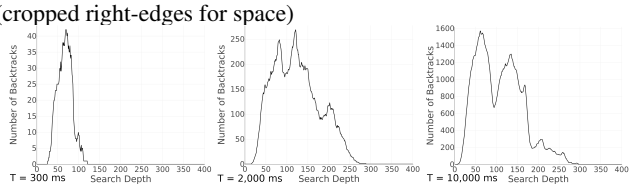


Figure 4: SHAPE regime of GAC on pseudo-aim-200-1-6-4 (cropped right-edges for space)

Of the above three regimes, SHAPE can recognize the most interesting changes in the BpD because it recognizes the change of the depth where thrashing occurs. Figures 3 and 4 show two regime progressions of GAC on the pseudo-aim-200-1-6-4 instance.[3] Figure 3 emphasizes the growth over time, while Figure 4 highlights the change in shape of the BpD.

We generate animations (of a user-specified duration) by applying time dilation on the histories generated with any of the three above behaviors. Currently, WORMHOLE implements two time-dilation methods, namely, EQUAL and PROPORTIONAL. EQUAL assigns to each regime an equal amount of time, while PROPORTIONAL assigns to each regime an animation time that is proportional to the regime's duration in search. In addition, WORMHOLE allows the user to directly modify the percentage of time that each regime takes independently of the above two dilation methods.

5 Case Study: PREPEAK$^+$

In this section, we use WORMHOLE to understand and compare the behavior of PREPEAK$^+$, a new reactive strategy for enforcing high level consistency during search [Woodward *et al.*, 2018]. To this end, we solve the CSP instance pseudo-aim-200-1-6-4 studied in Section 4 with backtrack search under three settings: (1) maintaining GAC, (2) with APOAC, and (3) PREPEAK$^+$. PREPEAK$^+$ is conservative in that it primarily enforces GAC. However, it triggers an HLC, such as POAC, when the number of backtracks per depth (BpD) reaches a given threshold value θ but only when search backtracks to levels shallower than the depth where the threshold is met. PREPEAK$^+$ keeps firing the HLC as long as the BpD at the considered depth is smaller than θ. Furthermore, every time it backtracks, PREPEAK$^+$ updates the values of θ by reducing it or increasing it according to three geometric laws depending on whether the HLC yields wipeout (i.e., it is effective), filters the search space, or yields no filtering (i.e., the HLC calls are wasteful). WORMHOLE reports the costs of three search algorithms as shown in Table 2.

Figure 5 shows the BpD for GAC at the end of search.

[3]Instance pseudo-aim-200-1-6-4 of the benchmark pseudo-aim from `www.cril.univ-artois.fr/~lecoutre/benchmarks.html`.

Table 2: Solving instance pseudo-aim-200-1-6-4 with GAC, APOAC, PREPEAK$^+$

	GAC	APOAC	PREPEAK$^+$
CPU time (sec)	185,045	66,816	17,836
#NV	3,978,074	47,457	284,289
$\max_{\text{BpD}}$	34,023	407	2,421
#HLC calls		7,739	228

This curve exhibits a peak around depth 100 with $\max_{\text{BpD}} = 34,023$ showing that GAC is too weak to filter out bad values: it spends much of its time thrashing around this depth level.

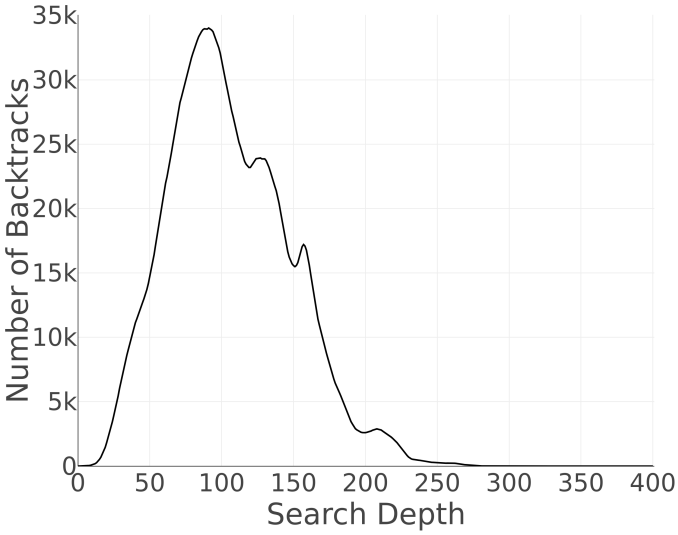


Figure 5: BpD and CpD of GAC on pseudo-aim-200-1-6-4

Figure 6 shows the BpD (black) and CpD (colored) curves for APOAC. Examining the BpD curve, we realize that

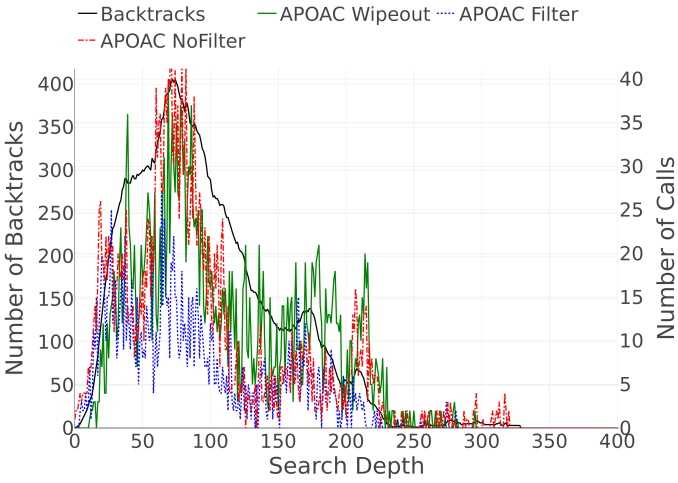


Figure 6: BpD (black) and CpD's (colored) of APOAC on pseudo-aim-200-1-6-4

APOAC so effectively prunes the 'bad subtrees' from the

search space that it dramatically reduces $\max_{\text{BpD}}$ down to 407 and the location of peak to around depth 75. We see that this instance benefits from enforcing an HLC such as POAC with a clear benefit on the CPU time (which is reduced by one order of magnitude from GAC). However, by observing the colored curves in Figure 6, we notice that the number of calls to POAC that are ineffective (red curve) are of the same order as those that yield wipeout (green curve). The detailed CpD curves hint to some savings that could be further obtained could one cancel the wasteful calls to POAC.

Figure 7 shows the BpD (black) and CpD (colored) curves for PREPEAK$^+$. PREPEAK$^+$ is conservative in that it calls

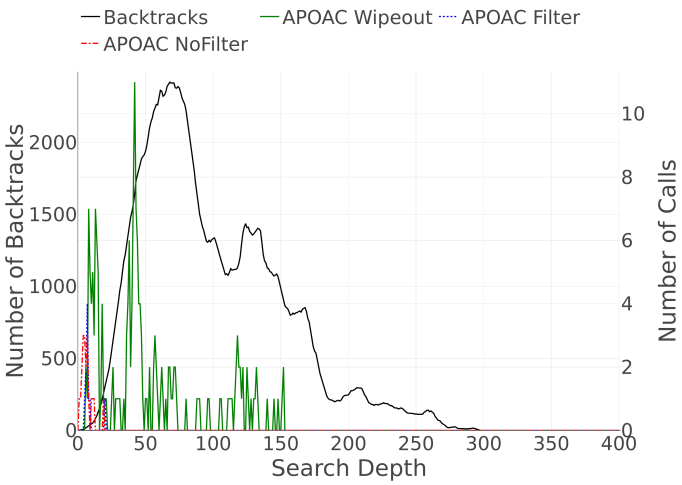


Figure 7: BpD (black) and CpD's (colored) of PREPEAK$^+$ on pseudo-aim-200-1-6-4

an HLC only when search thrashes, justifying the cost of a stronger but more costly consistency algorithm. Indeed, we observe that the peak value of BpD is smaller than for GAC but greater than for APOAC (2,421 versus 34,023 and 407, respectively). However, examining the detailed CpD curves shows that advantage of PREPEAK$^+$: Indeed, the wasteful calls to POAC (red) are almost eliminated and the total calls to POAC are reduced down to 228 for PREPEAK$^+$ from 7,739 for APOAC. This economy in the calls to POAC is immediately translated by the reduction of the CPU time. Thus, despite the fact that PREPEAK$^+$ explores a larger search tree than APOAC (see number of nodes visited) because it does not call the HLC at each variable instantiation, it effectively reacts to thrashing, calling the HLC only when it is needed, but spontaneously reverting to GAC otherwise.

This example illustrates the pertinence of the tools provided by WORMHOLE in visually explaining the behavior of search and the benefits of PREPEAK$^+$.

6 Architecture

Figure 8 shows the architecture of WORMHOLE. A constraint solver exports data to a generic, JSON-based log file that records the differences in tracked values over time (States 1-2). These logs are then loaded by the user into WORMHOLE, through (State 3), and parsed (State 4), into various data structures. At this point, the timeline data is pregenerated for

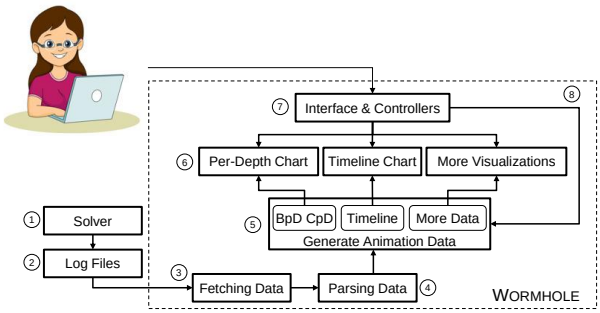


Figure 8: Architecture of WORMHOLE

immediate use while the BpD and CpD data are cached for further data generation (State 5). Timeline data is then sent immediately to the timeline chart and displayed to the user (State 6). Through the use of control panel interaction the user may generate new animations, select specific time information, or control their playback (State 7). The user input controller then calls (State 8) for the re-generation of data (State 5) which updates the animation data and sent to the user's view (State 6).

We designed WORMHOLE using web technologies of HTML, JavaScript, and CSS, based on the React-Redux model [Walke, ; Abramov and Clark,]. In this framework, React manages view rendering and user interaction, while Redux manages the application state. While standard web-application systems work adequately for logs with limited data (10's of MB), much search data is too large (100's of MB) to be handled by such systems. For the sake of performance, we enhance the system as follows.

JavaScript is a single-threaded language because it is based on an event-loop paradigm, which prevents the user from interacting with the application during periods of heavy computation. In WORMHOLE, we use Web Workers, a modern feature of the browser, to prevent blocking: it offloads most of the processing to a separate process. In this separate process, we perform initial data parsing, animation pre-fetching, and time-specific frame loading, which all are processor-intensive tasks.

To accelerate frame loading during animation, we pre-fetch all animation data in the data process and transfer it using JavaScript Transferables, which are data structures for rapidly transferring large data sets between web workers and the main thread [Mozilla and individual contributors,]. Further, when parsing the log, we perform regular sampling on the BpD and CpD data in order to reduce the work needed to calculate specific time frames because only data differences over time are stored in memory. When users request BpD and CpD data for a specific time, we perform nearest neighbor interpolation, only accepting samples of time not greater than the request before calculating with the rest of the difference data.

With the increasing size of the backtrack data, it is not viable to interactively visualize the entire backtrack data with limited memory size and rendering capacity. To address this issue, we apply a simple multi-resolution modeling technique that samples the backtrack data-set across multiple resolu-

tions and, for each resolution, partitions the data-set into blocks, each containing 1024 data points [Wynn *et al.*, 1997]. When users zoom on this chart to different resolutions, the blocks that have a resolution at least as great as the user's view and include points visible to the user are displayed on the chart. This optimization drastically increases performance on large searches from 0.2 frames per second (fps) to 20 fps.

7 Conclusions

WORMHOLE offers a number of new visualization and animation techniques that allow the user to explore and understand the behavior of backtrack search and compare the performance of different algorithms. We are currently applying our techniques to Choco, a popular constraint solver that uses binary search (2-way branching) while exploring new functionalities.

Acknowledgments

This research is supported by NSF Grant No. RI-1619344 and NSF CAREER Award No. III-1652846. Robert Woodward was supported by an NSF GRF Grant No. 1041000 and a Chateaubriand Fellowship. The experiments were completed utilizing the Holland Computing Center of the University of Nebraska, which receives support from the Nebraska Research Initiative.

References

[Abramov and Clark,] Dan Abramov and Andrew Clark. Redux. `https://redux.js.org/`. Accessed: 2017-04-21.

[Balafrej *et al.*, 2013] Amine Balafrej, Christian Bessiere, Remi Coletta, and El-Houssine Bouyakhf. Adaptive Parameterized Consistency. In *Proc. of CP 2013*, volume 8124 of *LNCS*, pages 143–158. Springer, 2013.

[Balafrej *et al.*, 2014] Amine Balafrej, Christian Bessiere, El-Houssine Bouyakhf, and Gilles Trombettoni. Adaptive Singleton-Based Consistencies. In *Proc. of AAAI 2014*, pages 2601–2607, 2014.

[Balafrej *et al.*, 2015] Amine Balafrej, Christian Bessière, and Anastasia Paparrizou. Multi-Armed Bandits for Adaptive Constraint Propagation. In *Proc. of IJCAI 2015*, pages 290–296, 2015.

[Bennaceur and Affane, 2001] Hachemi Bennaceur and Mohamed-Salah Affane. Partition-k-AC: An Efficient Filtering Technique Combining Domain Partition and Arc Consistency. In *Proceedings of* 7^{th} *International Conference on Principle and Practice of Constraint Programming (CP'01)*, volume 2239 of *LNCS*, pages 560–564. Springer, 2001.

[Bessière *et al.*, 2008] Christian Bessière, Kostas Stergiou, and Toby Walsh. Domain Filtering Consistencies for Non-Binary Constraints. *Artificial Intelligence*, 172:800–822, 2008.

[Borrett *et al.*, 1996] James E. Borrett, Edward P.K. Tsang, and Natasha R. Walsh. Adaptive Constraint Satisfaction: The Quickest First Principle. In *Proc. of ECAI 1996*, pages 160–164, 1996.

[Boussemart *et al.*, 2004] Frédéric Boussemart, Fred Hemery, Christophe Lecoutre, and Lakhdar Sais. Boosting Systematic Search by Weighting Constraints. In *Proc. of ECAI 2004*, pages 146–150, 2004.

[Carro and Hermenegildo, 2000] Manuel Carro and Manuel Hermenegildo. Tools for Constraint Visualisation: The VIFID/TRIFID Tool. In *Analysis and Visualization Tools for Constraint Programming: Constraint Debugging*, volume 1870 of *Lecture Notes in Computer Science*, pages 253–272. Springer, 2000.

[Debruyne and Bessiere, 1997] Romuald Debruyne and Christian Bessiere. Some Practicable Filtering Techniques for the Constraint Satisfaction Problem. In *Proc. of IJCAI 1997*, pages 412–417, 1997.

[Eén and Biere, 2005] Niklas Eén and Armin Biere. Effective Preprocessing in SAT Through Variable and Clause Elimination. In *Proc. of SAT 2005*, volume 3569 of *LNCS*, pages 61–75. Springer, 2005.

[Epstein *et al.*, 2005] Susan L. Epstein, Eugene C. Freuder, Richard M. Wallace, and Xingjian Li. Learning Propagation Policies. In *International Workshop on Constraint Propagation and Implementation*, pages 1–15, 2005.

[Freuder and Elfe, 1996] Eugene C. Freuder and Charles D. Elfe. Neighborhood Inverse Consistency Preprocessing. In *Proc. of AAAI 1996*, pages 202–208, 1996.

[Freuder and Wallace, 1991] Eugene C. Freuder and Richard J. Wallace. Selective Relaxation For Constraint Satisfaction Problems. In *Proc. of ICTAI 1991*, pages 332–339, 1991.

[Hayes, 1990] Patrick J. Hayes. *Readings in Qualitative Reasoning About Physical Systems*, chapter The Second Naive Physics Manifesto, pages 46–63. Morgan Kaufmann, 1990.

[Järvisalo *et al.*, 2012] Matti Järvisalo, Marijn J. H. Heule, and Armin Biere. Inprocessing Rules. In *Proc. of IJCAR 2012*, volume 7364 of *LNCS*, pages 355–370. Springer, 2012.

[Kuipers, 1994] Benjamin Kuipers. *Qualitative Reasoning - Modeling and Simulation with Incomplete Knowledge*. MIT Press, 1994.

[Lagerkvist and Schulte, 2009] Mikael Z. Lagerkvist and Christian Schulte. Propagator Groups. In *Proceedings of* 5^{th} *International Conference on Principle and Practice of Constraint Programming (CP'99)*, volume 5732 of *LNCS*, pages 524–538. Springer, 2009.

[Mackworth, 1977] Alan K. Mackworth. Consistency in Networks of Relations. *Artificial Intelligence*, 8:99–118, 1977.

[Mozilla and individual contributors,] Mozilla and individual contributors. MDN Web Docs: Transferable. `https://developer.mozilla.org/en-US/docs/Web/API/Transferable`. Accessed: 2017-04-21.

[Paparrizou and Stergiou, 2012] Anastasia Paparrizou and Kostas Stergiou. Evaluating Simple Fully Automated Heuristics for Adaptive Constraint Propagation. In *Proc. of ICTAI 2012*, pages 880–885, 2012.

[Paparrizou and Stergiou, 2017] Anastasia Paparrizou and Kostas Stergiou. On Neighborhood Singleton Consistencies. In *Proc. of IJCAI 2017*, pages 736–742, 2017.

[Rényi, 1961] Alfréd Rényi. On Measures of Entropy and Information. Technical report, Hungarian Academy of Sciences, Budapest, Hungary, 1961.

[Samaras and Stergiou, 2005] Nikos Samaras and Kostas Stergiou. Binary Encodings of Non-binary Constraint Satisfaction Problems: Algorithms and Experimental Results. *JAIR*, 24:641–684, 2005.

[Schulte *et al.*, 2015] Christian Schulte, Guido Tack, and Mikael Z. Lagerkvist. *Modeling and programming with Gecode*, 2015.

[Schulte, 1996] Christian Schulte. Oz Explorer: A visual constraint programming tool. In *International Symposium on Programming Language Implementation and Logic Programming*, pages 477–478. Springer, 1996.

[Shishmarev *et al.*, 2016] Maxim Shishmarev, Christopher Mears, Guido Tack, and Maria Garcia de la Banda. Visual Search Tree Profiling. *Constraints*, 21(1):77–94, 2016.

[Simonis and Aggoun, 2000] Helmut Simonis and Abder Aggoun. Search-Tree Visualisation. In *Analysis and Visualization Tools for Constraint Programming: Constraint Debugging*, volume 1870 of *Lecture Notes in Computer Science*, pages 191–208. Springer, 2000.

[Simonis and O'Sullivan, 2011] Helmut Simonis and Barry O'Sullivan. Almost Square Packing. In *Proc. of CPAIOR 2011*, pages 196–209. Springer, 2011.

[Simonis *et al.*, 2010] Helmut Simonis, Paul Davern, Jacob Feldman, Deepak Mehta, Luis Quesada, and Mats Carlsson. A Generic Visualization Platform for CP. In *Proc. of CP 2010*, pages 460–474, 2010.

[Stergiou, 2008] Kostas Stergiou. Heuristics for Dynamically Adapting Propagation. In *Proc. of ECAI 2008*, pages 485–489, 2008.

[Walke,] Jordan Walke. React. `https://reactjs.org/`. Accessed: 2017-04-21.

[Wallace, 2015] Richard J. Wallace. SAC and Neighbourhood SAC. *AI Communications*, 28(2):345–364, 2015.

[Woodward *et al.*, 2011] Robert Woodward, Shant Karakashian, Berthe Y. Choueiry, and Christian Bessiere. Solving Difficult CSPs with Relational Neighborhood Inverse Consistency. In *Proc. of AAAI 2011*, pages 112–119, 2011.

[Woodward *et al.*, 2014] Robert J. Woodward, Anthony Schneider, Berthe Y. Choueiry, and Christian Bessiere. Adaptive Parameterized Consistency for Non-Binary CSPs by Counting Supports. In *Proc. of CP 2014*, volume 8656 of *LNCS*, pages 755–764. Springer, 2014.

[Woodward *et al.*, 2018] Robert J. Woodward, Berthe Y. Choueiry, and Christian Bessiere. A Reactive Strategy for High-Level Consistency During Search. In *Proc. of IJCAI 2018*, pages 1–8, 2018.

[Wotzlaw *et al.*, 2013] Andreas Wotzlaw, Alexander van der Grinten, and Ewald Speckenmeyer. Effectiveness of Pre- and Inprocessing for CDCL-based SAT Solving. Technical report, Institut für Informatik, Universität zu Köln, Germany, 2013.

[Wynn *et al.*, 1997] William C. Wynn, J. Fritz Barnes, Bernd Hamann, and Mark Miller. Multiresolution and Adaptive Rendering Techniques for Structured, Curvilinear Data. In *Scientific Visualization Conference (dagstuhl '97)*, pages 332–332, 1997.

Toward Learning Finite State Representations of Recurrent Policy Networks

Anurag Koul[1], **Sam Greydanus**[1], **Alan Fern**[1]
[1] Oregon State University
{koula, greydasa, afern}@oregonstate.edu

Abstract

Recurrent neural networks (RNNs) are an effective representation of control policies for a wide range of reinforcement learning problems. RNN policies, however, are particularly difficult to explain and understand due to their use of continuous-valued internal memory. In this paper, we study an approach for bringing us closer to understanding these policies based on learning accurate finite-state representations of the continuous-valued memory. In particular, we learn policies represented as Moore Machine Networks (MMNs), where the memory is composed of binary variables that ideally provide a more interpretable view of how the memory is used. We describe a learning approach based on training Binary Bottleneck Networks (BBNs) to encode the continuous RNN state and then inserting the BBN directly in the RNN, which then serves as a compact binary form of memory. We present preliminary empirical results on several types of synthetic learning environments, which shows the potential for this approach to learn MMNs and provide insight into the memory usage of RNN policies.

1 Introduction

Deep reinforcement learning (RL) and imitation learning (IL) have demonstrated impressive performance across a wide range of applications. Unfortunately, the learned policies are difficult to understand and explain, which limits the degree that they can be trusted and used in high-stakes applications. Such explanations are particularly problematic for policies represented as recurrent neural networks (RNNs) [Mnih *et al.*, 2016; Mikolov *et al.*, 2010], which are increasingly used to achieve state-of-the-art performance [Mnih *et al.*, 2015; Silver *et al.*, 2017]. This is because RNN policies use internal memory to encode features of the observation history, which are critical to their decision making, but extremely difficult to interpret. In this paper, we take a step towards comprehending and explaining RNN policies by learning more compact memory representations.

Explaining RNN memory is challenging due to the typical use of high-dimensional continuous memory vectors that are updated through complex gating networks (e.g. LSTMs, GRUs [Hochreiter and Schmidhuber, 1997; Chung *et al.*, 2014; Cho *et al.*, 2014]). We hypothesize that, in many cases, the continuous memory is capturing and updating one or more discrete concepts. If exposed, such concepts could significantly aid explainability. This motivates learning memory representations based on a moderate number of binary variables. In this case, understanding the memory can be approached by understanding the role of each memory bit. Of course, not all RNN policies will have compact binary representations, but many powerful forms of memory usage can be captured in this way.

Our main contribution is to introduce an approach for transforming an RNN policy with continuous memory and continuous observations to a finite-state representation known as a Moore Machine. To accomplish this we introduce the idea of *Binary Bottleneck Network (BBN) insertion*. BBNs are simply auto-encoders, where the latent representation is binary. Given a trained RNN, we train BBNs to encode the memory states and observation vectors that are encountered during the RNN operation. We then insert the BBNs into the trained RNN policy in place of the "wires" that propagated the memory and observation vectors. The combination of the RNN and BBN results in a policy represented as a Moore Machine Network (MMN) with binary memory and observations and transitions controlled by the continuous RNN and BBN elements. The MMN can be used directly or fine-tuned to improve on inaccuracies introduced by BBN insertion.

While training binary network is often considered to be quite challenging, we show that a simple approach works well in the case of BBNs. In particular, we demonstrate that "straight through" gradient estimators (e.g. [Bengio *et al.*, 2013; Courbariaux *et al.*, 2016]) are quite effective. We also demonstrate the approach of BBN insertion is more effective than simply learning a binary memory RNN from scratch. This is an interesting example of where training a more complex, continuous model can provide guidance for arriving at a discrete model, which would have been difficult to learn without the guidance.

We focus our preliminary experimental work on environments where we know the ground truth minimal state-space policy. In particular, we define a parameterized class of environments where the complexity of memory usage can be varied along multiple dimensions. We show that our approach is able to effectively extract the ground truth policies, or close

approximations, for several policies that use memory in different ways.

2 Related Work

There have been efforts made in the past to understand the internals of Recurrent Networks (e.g. [Karpathy *et al.*, 2015; Arras *et al.*, 2017; Strobelt *et al.*, 2016]). However, to the best of our knowledge there is no prior work on learning binary-memory representations of continuous RNN policies. We are also unaware of work on learning binary-memory policy networks from scratch. Our work, however, is related to a large body of work on learning finite-state representations of recurrent neural networks. Below we summarize the branches of that work and the relationship to our own.

There has been a significant history of work on extracting Finite State Machines (FSMs) from recurrent networks trained to recognize languages [Zeng *et al.*, 1993; Tiňo *et al.*, 1998; Cechin *et al.*, 2003]. Typical approaches include discretizing the continuous memory space via gridding or clustering followed by minimization. A more recent approach is to use classic query-based learning algorithms to extract FSMs by asking membership and equivalence queries [Weiss *et al.*, 2017]. However, none of these approaches directly apply to learning policies, which require extending to Moore Machines. In addition, all of these approaches produce "flat state-space" FSM, where the states are atomic entities with no explicit factors. In contrast, we consider learning Moore Machine Networks where the states are factored into a set of binary variables. This has the potential to aid in interpretability, since one can consider interpreting the variables that compose the state-space rather than interpreting each individual state.

The most closely related work is to ours also focused on learning FSMs [Zeng *et al.*, 1993]. However, the approach is based on directly learning recurrent networks with binary memory, which are qualitatively similar to the memory representation of our MMNs. That work, however, focused on learning from scratch rather than aiming to describe the behavior of a continuous RNN. Our work extends that approach to learn MMNs and more importantly introduces the method of BBN insertion as a way of learning via guidance from a continuous RNN. Our experiments illustrate that learning binary memory networks from scratch can be much more difficult than leveraging guidance from an RNN.

We note that there has been prior work on learning fully binary networks, where the activation functions and/or weights are binary (e.g. [Bengio *et al.*, 2013; Courbariaux *et al.*, 2016; Hinton, 2012]). The goal of that line of work is typically to learn more time and space efficient networks. Rather, we focus on learning only binary representations of memory and observations, while allowing the rest of the network to use arbitrary activations and weights. This is due to our alternative goal of supporting interpretability rather than efficiency.

3 Recurrent Policy Networks: Continuous and Binary

Continuous Recurrent Network (CRN) policies are commonly used in reinforcement learning to represent policies that require internal memory. At each time step, a CRN is given an observation o_t (e.g. image) and must output an action a_t to be taken in the environment. During execution a CRN maintains a continuous-valued hidden state h_t, which is updated on each transition and influences the action choice. In particular, given the current observation o_t and current state h_t, a CRN performs the following operations: 1) Extract a set of observation features f_t from o_t, for example, using a CNN when observations are images, 2) Outputting an action $a_t = \pi(h_t)$ according to policy π, which is often a linear softmax function of h_t, 3) transition to a new state $h_{t+1} = \delta(f_t, h_t)$ where δ is the transition function, which is often implemented via different types of gating networks such as LSTMs or GRUs.

The continuous and high dimensional nature of h_t and f_t in a CRN can make interpreting the role of memory difficult. This motivates our goal of extracting compact binary representations of h_t and f_t. Such representations have the potential to allow investigating the role of a manageable number of individual bits that describe key features of the memory and observations. For this purpose we introduce Moore Machines and their deep network counterparts.

A classical Moore Machine is a standard finite state machine where all states are labeled by output values, which in our case will correspond to actions. In particular, a Moore Machine is described by a finite set of (hidden) states $\tilde{H}$, an initial hidden state $\tilde{h}_0$, a finite set of observations $\tilde{O}$, a finite set of actions A, a transition function δ, and a policy π that maps hidden states to actions. The transition function $\tilde{\delta} : \tilde{H} \times \tilde{O} \rightarrow \tilde{H}$ returns the next hidden state $\tilde{\delta}(\tilde{h}_t, \tilde{o}_t)$ given the current state $\tilde{h}_t$ and observation $\tilde{o}_t$. By convention we will use h_t and $\tilde{h}_t$ to denote continuous and finite states respectively and similarly for other quantities and functions.

A classic Moore Machine operates by starting in the initial state $\tilde{h}_0$ and then at each time step when in state $\tilde{h}_t$, the machine outputs action $\tilde{\pi}(\tilde{h}_t)$, observes the next observation $\tilde{o}_{t+1}$ and then transitions to state $\tilde{h}_{t+1} = \tilde{\delta}(\tilde{h}_t, \tilde{o}_t)$. Such Moore Machines can model complex policies with internal state, but constrain the hidden state and observations into discrete finite sets.

A Moore Machine Network (MMN) is a Moore Machine where the transition function $\tilde{\delta}$ and policy $\tilde{\pi}$ are represented via deep networks. In addition, since often the raw observations given to an MMN will often be continuous, or from an effectively unbounded set (e.g. images), an MMN will also provide a mapping $\tilde{\theta}$ from the continuous observations to a finite discrete observation space $\tilde{O}$. Here $\tilde{\theta}$ will also be represented via a deep network. In this work, we consider binary state and observation representations where each $\tilde{h} \in \tilde{H}$ is a binary bit vector and each discrete observation in $\tilde{O}$ is a binary feature vector $\tilde{f}$ that describes the raw observation.

Based on the above discussion, an MMN can be viewed as a traditional CRN, where: 1) The memory is restricted to be composed of binary activation units, and 2) The environmental observations are transformed to a binary representation $\tilde{f}$ before being fed to the CRN. Given an approach for incorporating binary units into the backpropagation process, it is straightforward in concept to attempt to learn MMNs from scratch via standard CRN learning algorithms. However, we

have found that learning such MMNs from scratch can be quite difficult for non-trivial problems. This is true even when learning a CRN for the problem is relatively easy. This motivates the goal of learning MMNs by first learning a CRN, which is then used to guide the extraction of an equivalent MMN. Below we introduce a new approach for doing this based on the idea of binary bottleneck insertion.

4 Learning Moore Machine Networks from CRNs

Given a trained CRN R, our key idea is to first learn *binary bottleneck networks (BBNs)* for embedding the continuous features and states of R into binary spaces. We will then insert the BBNs into R in such a way that its behavior is minimally changed. The resulting network can be viewed as consuming binary features and maintaining binary state, which is effectively an MMN. Below we describe the steps in further detail, which are illustrated in Figure 1.

4.1 Binary Bottleneck Networks

A BBN is simply an autoencoder where the latent representation between the encoder and decoder (i.e. the bottleneck) is constrained to be composed of binary activation units. While, traditional autoencoders are generally used for the purpose of dimensionality reduction in continous space [Hinton and Salakhutdinov, 2006], BBNs are motivated by the goal of discretizing a continous space. Conceptually, this can be done by quantizing the activations of units in the encoding layer.

We represent a BBN via a continuous multilayer encoder E, which maps inputs x to a latent encoding $E(x)$, and a corresponding multilayer decoder D. To binarize the encoding the BBN output is given by

$$b(x) = D(\text{sign}(E(x)))$$

where the sign function maps each vector component to $+1$ or -1 depending on whether the component is non-negative or negative respectively. To train a BBN we optimize the standard L_2 reconstruction error for a given input x:

$$L(x) = \|x - b(x)\|^2$$

Importantly the inclusion of the sign function in the BBN effectively allows us to view the last layer of E as producing a binary encoding. Here we use $+1$ and -1 to encode binary values, which has been a common choice in prior work on learning binary network representations (e.g. [Courbariaux *et al.*, 2016]).

Of course introducing the sign function in the BBN results in $b(x)$ being non-differentiable, making it apparently incompatible with backpropagation, since the gradients between the decoder and encoder will almost always be zero. While there are a variety of ways to deal with this issue, we have found that the straight-through estimator, as suggested and used in prior work [Hinton, 2012; Bengio *et al.*, 2013; Courbariaux *et al.*, 2016] is quite effective. In particular, the standard straight-through estimator of the gradient simply treats the sign function as the identity function during back-propagation. In our implementation, similarly to prior work [Courbariaux *et al.*, 2016], we clip the gradient signals through the sign units to $+1$ and -1, which can be viewed as propagating the gradients through the hard tanh function $\text{Htanh}(x) = \max(-1, \min(1, x))$ as suggested in [Courbariaux *et al.*, 2016] which otherwise could lead to unstable performance when gradients become too large.

4.2 Bottleneck Insertion

Given a CRN policy we can run the policy in the target environment in order to produce an arbitrarily large set of training sequences of triples (o_t, f_t, h_t), giving the observation, corresponding CRN features, and hidden state at time t. Let F and H be the sets of all observed features and states respectively. The first step of our approach is to train two BBNs, b_f and b_h, on F and H respectively. If the BBNs are able to achieve low reconstruction error then the binary bottlenecks can be viewed as providing high-quality binary encodings of the original states and features.

We now view b_f and b_h as "wires" that propagate the input to the output, with some noise due to imperfect reconstruction. We insert these wires into the original CRN in the natural way (see Figure 1). The b_f BBN is inserted between the CRN units that compute the features f and the nodes those units are connected to. The b_h BBN is inserted between the output of the recurrent network block and the input to the recurrent block. If b_f and b_h always produced perfect reconstructions, then the result of inserting them as described would not change the behavior of the CRN. Yet, the CRN can now be viewed as an MMN since the bottlenecks of b_f and b_h provide a binary representation of the features $\tilde{f}_t$ and states $\tilde{h}_t$.

In practice, the BBNs will not achieve perfect reconstruction and thus, the resulting MMN may not behave identically to the original CRN. In this preliminary investigation, We found that the performance of the resulting MMN is very close to the CRN directly after insertion. However, when there is non-trivial performance degradation, we can fine-tune the MMN by training on the original data of the CRN. Importantly, since our primary goal is to learn a representation of the original CRN, during fine-tuning our objective is to have the MMN match the softmax distribution over actions produced by the CRN. We found that training in this way was significantly more stable than training the MMN to simply output the same action as the CRN.

4.3 Moore Machine Extraction and Minimization

After obtaining the MMN, one could use visualization and other analysis tools to investigate the memory and it's feature bits in order to gain a semantic understanding of their roles. Solving the full interpretation problem in a primitive way is beyond the scope of this work. We view this as a critical research direction for future work, not only for MMNs but for deep networks in general.

Another way to gain insight is to use the MMN to produce a minimal equivalent Moore Machine over a flat state space. This machine can be analyzed to understand the role of different machine states and how they are related. To create the Moore Machine we run the learned MMN to produce

a dataset of $< \tilde{h}_{t-1}, \tilde{f}_t, \tilde{h}_t, a_t >$, giving the consecutive pairs of binary states, the binary features that led to the transition, and the action selected after the transition. The states of the Moore Machine will correspond to the p distinct binary states in the data and the observations of the machine will be the q unique binary feature vectors in the data. The transition function of the machine δ is constructed from the data by producing a $p \times q$ transaction table that captures the transitions seen in the data. In general, the number of states p will not be minimal. Thus, we then apply standard Moore Machine minimization techniques to arrive at the minimal equivalent Moore Machine.

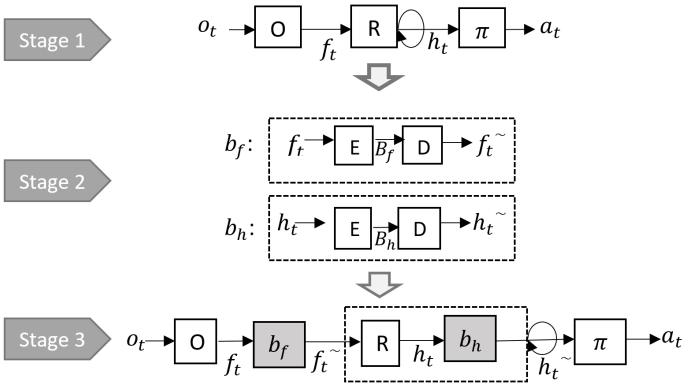


Figure 1: Learning Moore Machine Networks. (1.) Learn a CRN policy. (2.) Learn BBN's. (3.) Insertion and Fine Tuning. O, R are feature-extraction and recurrent modules, respectively

5 Experimental Setup

Our experiments address the following question: 1) Is it possible to extract MMNs from CRNs without significant loss in Performance? 2) Is the bottleneck insertion approach a more effective way to learn MMNs compared to training MMNs from scratch? 3) Do the learned MMNs help with interpretability of the recurrent policies? We addressed these questions by considering a parameterized synthetic environment for learning policies where the optimal policies are known along with their minimal Moore Machine representation. Compared to evaluating in domains where there is no information about the true underlying machines, these controlled environments with known ground truth allows for a more rigorous analysis of extraction accuracy.

Below we first describe the general experimental environment along with the specific instantiations that we consider in the experiments. Next, we describe some implementation details of our system.

5.1 Experimental Environment

We have developed a parametric POMDP domain called GoldRush. The goal is for different parameterizations of the domain to place qualitatively different requirements on how a recurrent policy must use memory to be successful.

GoldRush is an environment with a single agent that needs to collect gold at every time-step to survive and the objective is to maximize the survival time. The underlying states of the environment are defined by M modes of operation. At each time step the environment is in a particular mode and the full environment state is a pair (m_t, c_t) where $m_t \in \{1, 2, \ldots, M\}$ is the mode at time t and c_t is the count of time-steps that the system has been consecutively in mode m_t. Each mode is parameterized by a life-span $ls(m)$, which indicates how long the system will be in the particular mode. When the counter c_t reaches the life-span , the next mode is generated according to some distribution. Based on the state (m_t, c_t) the environment outputs an observation o_t as per some distribution (see below). For each such state, there is a specific action a_t that collects gold, while all other actions fail to collect gold. In our experiments, the correct action a_t depends only on the mode m_t, though in general it could also depend on the remaining life of the mode indicated by c_t.

The state (m_t, c_t) of the environment is hidden from the learning agent, which means the state must be inferred from observations. In our current environment, the observation o_t is a continuous value uniformly sampled in $[i/M, (i+1)/M)]$ where i is uniformly sampled in $[0, M)$ on each time step, except on the first time step of a new mode. On the first time step, i is set equal to the true mode. This means, during the life-span of a mode m_t the observations o_t doesn't need to necessarily belong to the active mode m_t. It is only on the first step that the mode can be inferred, which makes the learning problem quite challenging. When the current mode reaches its life-span as indicated by c_t the next mode is chosen uniformly at random and c_t is reset to zero. The reward at each step is 1 if the correct action is selected for the mode and otherwise it is zero. Also, m_0 is set to the mode of first randomly sampled observation o_0

We conduct experiments with the following three instantiations of the GoldRush environment [1]:

- **Reader.** The first environment considers a situation where no memory is required. This test the ability of our system to train a recurrent policy on such an environment, extract an MMN, and then see if it is possible to determine that the recurrent policy does not actually use memory. This instantiation is arrived at by setting $ls(m) = 1$ for all modes.
- **Blind.** Contrary to above, this variation can be solved by a policy that only uses memory and does not need to consult observations. That is, the policy is an open-loop controller with memory. This allows us to test whether the extraction of an MMN could infer that a recurrent memory does not use observations at all. This type of behavior is sometimes observed in situations where a learned policy is overfit to the training experience, where it was possible to blindly take a sequence of actions. For example, this type of behavior is sometimes observed in policies learned by Atari agents. To implement this instantiation we define the transition function between modes to be deterministic rather than uniformly random and start each episode from a fixed start state.

[1] Implementation available @ `https://github.com/koulanurag/gym_x`. Also, ground truth minimal moore machine available @ `https://github.com/koulanurag/gym_x/blob/master/docs/README.md`

- **Sneak.** This environment requires both paying attention to observations in order to infer the mode as well as using memory to keep track of how long the system has been in a mode, which is critical for selecting the correct actions. This is the most general instance of the environment and can result in difficult problems when the number of modes and their life-spans grow.

5.2 Implementation Details

We use a number of modes equal to $M = 4$ in our experiments and the life-span for mode m_i is $ls(i) = ((i\%2) + 2)$. We restrict the maximum number of steps in an episode to be 300. We train a CRN on each environment using the same architecture. In particular, the CRN has a single feed-forward layer with four RELU units; followed by a GRU cell having 5 hidden-features and lastly a feed-forward layer having m neurons for each of the actions (there is one correct action per mode, noting that the mode is not exposed to the network). Also, both of the BBNs b^f and b^h have the same architecture of 1 feed-forward layer with 5 neurons and RELU activation followed by the encoding layer having B binary units in E. The decoder D for both of them also has the same architecture of 5 RELU units in the first layer followed by a linear layer with the same dimension as the BBN input. We refer to encoding layer of b_f and b_h as B_f and B_h; respectively.

In order to understand effects of binarization, we consider $B_f \in \{4, 8\}$ and $B_h \in \{10, 15, 20\}$. We use the Adam optimizer with $lr = 0.001$ to learn the BBNs. For fine-tuning we use Adam with $lr = 0.0001$, where we found that the reduced learning rate is necessary to stabilize learning. Our implementation [2] of MMN is based in PyTorch [Paszke *et al.*, 2017]

Table 1: Experimental Environments

Environment	States	Observation	Max. Reward Threshold
Reader	4	4	300
Blind	10	4	300
Sneak	10	4	300

Table 2: CRN Training

Environment	Epochs	Optimal Performance
Reader	3	300
Blind	14	300
Sneak	60	300

6 Empirical Results

Table-1 gives the details of each environment along with the reward of the optimal policy and the performance of trained CRNs is shared in Table-2. We see that the trained CRNs learn very effectively in a small number of iterations.

Our results are summarized in Tables-[2,3,4,5,6]. To understand the quality of our results; we trained each of our variants till optimal performance(via greedy policy) or to the point of saturation. We begin by presenting our results for conventional CRN architecture(stage 1 of our approach). This acts as a baseline in terms of performance and training-time.

Table 3: b_f Training

Env	B_f	Test-error
Reader	4	0.0017
	8	0.0003
Blind	4	0.0069
	8	0.0028
Sneak	4	0.0008
	8	0.0002

Table 4: b_h Training

Env	B_h	Test-error
Reader	10	0.0024
	15	0.0009
	20	0.001
Blind	10	0.0023
	15	0.002
	20	0.0016
Sneak	10	0.027
	15	0.015
	20	0.0107

BBN Reconstruction Performance. We first look at the reconstruction error of training the BBNs (stage-2) in Tables-[3,4]. We see that in all cases, the reconstruction error is quite small, which indicates that the straight-through estimator is an effective gradient approximation for these problems. It is important to note that training the BBNs is extremely fast compared to CRN training, since BBN training does not involve temporal dependencies. We found that since BBN are regression over real-valued vectors; this could lead to large weights in some cases. This is due to the need of reconstructing the numeric inputs from the much smaller binary bottleneck representation. These large weights can cause 'gradient' explosion in sub-sequent fine-tuning of the MMN, which we observed in practice. In order to cope with this, we performed gradient clipping in the [-1,1] range, which effectively addressed the issue.

BBN Insertion and Fine Tuning. In Table-5, we show results of embedding pre-trained BBNs into CRNs (stage 3 of our approach).The results comprehend the training time (Epochs) required to fine-tune the network after inserting b_f and b_h of respective encoding spaces. Overall, we observe that optimal performance could be restored in most of the cases with small fine-tuning especially with B_h encoding of 15 and 20 bits. There are cases where no fine tuning

[2] Code available @ `https://github.com/koulanurag/mmn`

($Epochs = 0$) was required as the bottleneck errors were minimal to cause any deviation in the greedy policy. In our case; considered variations for B_f didn't caused much performance degradation.

Moore Machine Extraction. In Table-5, we report the best performance, extracted observation-features (obs), state-space ($states$) sizes. On comparison with the ground-truth values in Table-1; we see that extraction hasn't given minimal representation. Performing minimization [4.3] over it produce the minimal machines in all cases where the learned MM was perfect. This shows that the learned MMN was somewhat inefficient in its memory use, but learned an equivalent machine to the ground truth. In cases where the learned MMN was not accurate (the '-' entries in the table) we found the number of MMN states was inconsistently very large and minimizing the resulting machines did not significantly reduce the number of states.

Training MMNs from Scratch. The above results are for pre-trained BBN being inserted into the trained CRNs. We now consider whether the pre-training was necessary, or if it would have been possible to train the combination of untrained CRN and (untrained BBN) from scratch. As shown in Table- 6, we found out that it was still feasible to train them directly in some cases. However; generally they took much longer to train then the suggested approach. Also, they are more prone to getting stuck in local optima as compared to the previous approach[6].

Interpretablity. We demonstrated that for the considered environments, it is possible to extract accurate MMNs from CRNs. Our eventual goal is to develop visualization tools that will allow for attaching semantic meanings to the bits of the MMN, which will provide insight into the memory usage. However, currently, we focus on using the extracted flat-state Moore Machines to provide some insight into the original CRN. In particular, it is straightforward to perform an automated analysis that shows states of the Moore Machine. Based on this analysis; we could identify when a memory was being used in the policy and when it was not used. For the case of 'Blind'; MMN learned to map entire observation space as a single feature vector implying that the learned policy relies solely on the memory. In general; whenever a Moore Machine from Recurrent Neural Network memory was used; the states had a deterministic transaction for all observation features.This analysis allowed us to distinguish between the three memory use-cases in our experiments[3]. We assume this would have been difficult to infer by just analyzing agent environment interaction.

Table 5: MMN Training - Pre-trained CRN and BBN

Env	B_f	B_h	Epoch	Best Reward	obs.	states	min. states	min. obs.
Reader	4	10	7	300	4	9	**4**	**4**
		15	2	300	4	18	**4**	**4**
		20	2	300	4	20	**4**	**4**
	8	10	4	300	5	11	**4**	**4**
		15	0	300	8	23	**4**	**4**
		20	0	300	8	33	**4**	**4**
Blind	4	10	45	220	-	-	-	-
		15	6	300	5	57	**10**	**1**
		20	0	300	3	95	**10**	**1**
	8	10	42	223	-	-	-	-
		15	3	300	9	51	**10**	**1**
		20	0	300	7	18	**10**	**1**
Sneak	4	10	62	185	-	-	-	-
		15	3	300	5	61	**10**	**4**
		20	4	300	5	101	**10**	**4**
	8	10	30	191	-	-	-	-
		15	2	300	7	65	**10**	**4**
		20	6	300	5	67	**10**	**4**

Table 6: MMN Training - Un-trained CRN and BBN

Env	B_f	B_h	Epoch	Best Reward	obs.	states	min. states	min. obs.
Reader	4	10	13	300	4	9	**4**	**4**
		15	30	300	5	9	**4**	**4**
		20	12	300	4	10	**4**	**4**
	8	10	3	300	8	9	**4**	**4**
		15	5	300	9	15	**4**	**4**
		20	3	300	9	17	**4**	**4**
Blind	4	10	60	120	-	-	-	-
		15	60	93	-	-	-	-
		20	75	91	-	-	-	-
	8	10	60	93	-	-	-	-
		15	110	93	-	-	-	-
		20	51	90	-	-	-	-
Sneak	4	10	100	290	-	-	-	-
		15	62	280	-	-	-	-
		20	12	300	4	85	**10**	**4**
	8	10	75	62	-	-	-	-
		15	60	285	-	-	-	-
		20	15	300	5	83	**10**	**4**

7 Conclusion & Future Work

We introduced an approach of extracting finite state Moore Machines from any trained RNN policy. The key idea, bottleneck insertion, is to train binary bottleneck networks to produce binary encodings of continuous RNN memory and input features, and then insert those bottlenecks into the RNN. Preliminary results in three synthetic environments yielded promising results and showed that accurate Moore Machines can be extracted and that those machines can provide high-level insight into the memory usage of an RNN. Our overall goal is to move toward explaining memory usage of RNN policies. This work is just one step toward that goal. A key missing step is to be able to produce semantically meaningful interpretations of the binary representations that aid in explaining the operation of an RNN. One approach to this will be to develop interactive visualization tools that humans can use to assign "names" to the learned binary features. Another avenue will be to employ rule learning algorithms on the binary representations to encode the fundamental transitions rules that they obey.

[3] Visual representation of extracted and minimized moore machine available @ `https://github.com/koulanurag/mmn/tree/master/results`

Acknowledgment

This work is supported by DARPA under grant N66001-17-2-4030.

References

[Arras *et al.*, 2017] Leila Arras, Grégoire Montavon, Klaus-Robert Müller, and Wojciech Samek. Explaining recurrent neural network predictions in sentiment analysis. *arXiv preprint arXiv:1706.07206*, 2017.

[Bengio *et al.*, 2013] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

[Cechin *et al.*, 2003] Adelmo Luis Cechin, D Regina, P Simon, and K Stertz. State automata extraction from recurrent neural nets using k-means and fuzzy clustering. In *Chilean Computer Science Society, 2003. SCCC 2003. Proceedings. 23rd International Conference of the*, pages 73–78. IEEE, 2003.

[Cho *et al.*, 2014] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

[Chung *et al.*, 2014] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[Courbariaux *et al.*, 2016] Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or-1. *arXiv preprint arXiv:1602.02830*, 2016.

[Hinton and Salakhutdinov, 2006] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[Hinton, 2012] Geoffrey Hinton. Neural networks for machine learning. *video lectures*, 2012.

[Hochreiter and Schmidhuber, 1997] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[Karpathy *et al.*, 2015] Andrej Karpathy, Justin Johnson, and Li Fei-Fei. Visualizing and understanding recurrent networks. *arXiv preprint arXiv:1506.02078*, 2015.

[Mikolov *et al.*, 2010] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, and Sanjeev Khudanpur. Recurrent neural network based language model. *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association*, pages 1045–1048, 2010.

[Mnih *et al.*, 2015] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518, 2015.

[Mnih *et al.*, 2016] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937, 2016.

[Paszke *et al.*, 2017] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017.

[Silver *et al.*, 2017] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of Go without human knowledge. *Nature Publishing Group*, 550, 2017.

[Strobelt *et al.*, 2016] Hendrik Strobelt, Sebastian Gehrmann, Bernd Huber, Hanspeter Pfister, and Alexander M Rush. Visual analysis of hidden state dynamics in recurrent neural networks. *arXiv preprint arXiv:1606.07461*, 2016.

[Tiňo *et al.*, 1998] Peter Tiňo, Bill G Horne, C Lee Giles, and Pete C Collingwood. Finite state machines and recurrent neural networks—automata and dynamical systems approaches. In *Neural networks and pattern recognition*, pages 171–219. Elsevier, 1998.

[Weiss *et al.*, 2017] Gail Weiss, Yoav Goldberg, and Eran Yahav. Extracting automata from recurrent neural networks using queries and counterexamples. *arXiv preprint arXiv:1711.09576*, 2017.

[Zeng *et al.*, 1993] Zheng Zeng, Rodney M Goodman, and Padhraic Smyth. Learning finite state machines with self-clustering recurrent networks. *Neural Computation*, 5(6):976–990, 1993.

Interpretable Neuronal Circuit Policies for Reinforcement Learning Environments

Mathias Lechner[1*], **Ramin M. Hasani**[1*], **Radu Grosu**[1],
[1] Cyber Physical Systems (CPS) Group, TU Wien
(mathias.lechner, ramin.hasani, radu.grosu) @tuwien.ac.at
[*]Equal Contributions

Abstract

We propose an effective way to create interpretable control agents, by *re-purposing* the function of a biological neural circuit model, to govern simulated and real world reinforcement learning (RL) test-beds. We model the tap-withdrawal (TW) neural circuit of the nematode, *C. elegans*, a circuit responsible for the worm's reflexive response to external mechanical touch stimulations, and learn its synaptic and neuronal parameters as a policy for controlling basic RL tasks. We also autonomously park a real rover robot on a pre-defined trajectory, by deploying such neuronal circuit policies learned in a simulated environment. For reconfiguration of the *purpose* of the TW neural circuit, we adopt a search-based RL algorithm. We show that our neuronal policies perform as good as deep neural network policies with the advantage of realizing interpretable dynamics at the cell level.

1 Introduction

Through natural evolution, the nervous system of the nematode, *C. elegans*, structured a near optimal wiring diagram [White *et al.*, 1986]. Its stereotypic brain composed of 302 neurons connected through approximately 8000 chemical and electrical synapses [Chen *et al.*, 2006]. *C. elegans* exhibits distinct behavioral mechanisms to process complex chemical stimulations [Bargmann, 2006], avoid osmotic regions [Culotti and Russell, 1978], sleep [Nichols *et al.*, 2017], show adaptive behavior [Ardiel and Rankin, 2010], perform mechanosensation [Chalfie *et al.*, 1985b], and to control muscles [Wen *et al.*, 2012].

The functions of many neural circuits within its brain have been identified [Wicks and Rankin, 1995; Chalfie *et al.*, 1985a; Li *et al.*, 2012; Nichols *et al.*, 2017]. In particular, a neural circuit which is responsible for inducing a forward/backward locomotion reflex when the worm is mechanically exposed to touch stimulus on its body, has been well-characterized [Chalfie *et al.*, 1985a]. The circuit is called tap-withdrawal (TW) and it comprises 9 neuron classes which are wired together by means of chemical and electrical synapses. Synaptic polarities (either being excitatory or inhibitory) of the circuit have then been predicted, suggesting that the circuit realizes a competitive behavior between forward and backward reflexes, in presence of touch stimulations [Wicks and Rankin, 1995; Wicks *et al.*, 1996].

Behavior of the tap-withdrawal (TW) reflexive response is substantially similar to the control agent's reaction in some standard control settings such as the impulse response of a controller operating on an *Inverted Pendulum* [Widrow, 1964; Doya, 2000; Russell and Norvig, 2010], a controller acting on driving an under-powered car, to go up on a steep hill, known as the *Mountain Car* [Moore, 1990; Singh and Sutton, 1996], and a controller acting on the navigation of a rover robot that plans to go from point A to B, on a planned trajectory, with two control commands of angular and linear velocity.

We intend to take advantage of the similarity and reconfigure the synaptic and neuronal parameters of a deterministic dynamic model of the TW neural circuit, in each of the mentioned control settings. We use publicly available reinforcement learning toolkits, to evaluate the performance of our neuronal circuit policies. The environments include the inverted pendulum [Schulman *et al.*, 2017], the continuous mountain car of OpenAI Gym[1] and rllab [2], and the Cart-pole of rllab [Duan *et al.*, 2016]. In a real robotic setting, We also determine a control task for a rover robot to park autonomously in a specific parking spot, by a learned TW neuronal policy. For all three control challenges, we preserve the near-optimal wiring structure of the TW circuit and adopt a search-based reinforcement learning (RL) algorithm for synaptic parametrization of the network. The approach is named as *neuronal circuit policies*.

Our principle contribution in this work is to demonstrate the performance of a compact neuronal circuit model from the brain of the *C. elegans* worm, as an interpretable continuous time recurrent neural network, in standard control and RL settings. In our experimental evaluations, we demonstrate that our control agent can achieve the performance of the conventional and the state-of-the-art artificial intelligence (AI) control agents, by solving five RL tasks. We show how a learned neuronal circuit policy in a simulated environment, can be transferred to a real robotic environment. We also demonstrate that the function of the neurons in a learned neuronal network is interpretable.

[1] `https://github.com/openai/gym`
[2] `https://github.com/rllab/rllab`

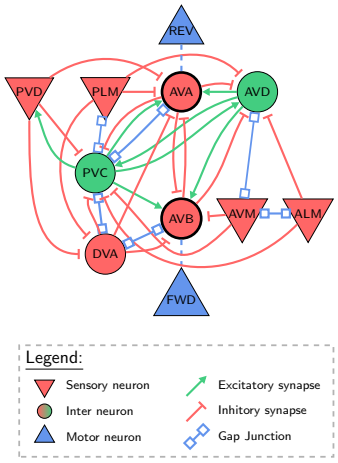


Figure 1: Tap-Withdrawal neural circuit schematic.

Algorithm 1 Random Search + Objective Indicator

Input: A stochastic objective indicator f,
a starting parameter θ
Input: Optimized parameter θ
$f_\theta \leftarrow f(\theta)$
for $k \leftarrow 1$ **to** maximum iterations **do**
  $\theta' \leftarrow \theta + rand()$
  $f_{\theta'} \leftarrow f(\theta')$
  if $f_{\theta'} < f_\theta$ **then**
    Set $\theta \leftarrow \theta'$
    $f_\theta \leftarrow f_{\theta'}$
    $i \leftarrow 0$
  end if
  $i \leftarrow i + 1$
  if $i > N$ **then**
    $f_\theta \leftarrow f(\theta)$
  end if
end for
return θ

2 Preliminaries

In this section, we first briefly describe the structure and dynamics of the tap-withdrawal neural circuit. We then introduce the mathematical neuron and synapse models utilized to build up the model of the circuit.

Tap-Withdrawal Neural Circuit Revisit – A mechanically exposed stimulus (i.e. tap) to the petri dish in which the worm inhabits, results in the animal's reflexive response in the form of a forward or backward movement. This response has been named as the *tap-withdrawal reflex*, and the circuit identified to underly such behavior is known as the *tap-withdrawal* (TW) neural circuit [Rankin *et al.*, 1990]. The circuit is shown in Figure 1. It is composed of four sensory neurons, PVD and PLM (posterior touch sensors), AVM and ALM (anterior touch sensors), four interneuron classes (AVD, PVC, AVA and AVB), and two subgroup of motor neurons which are abstracted as a forward locomotory neurons, FWD, and backward locomotory neurons, REV. Neurons recurrently synapse into each other with excitatory and inhibitory synaptic links. Throughout the paper, we illustrate how such recurrent neuronal network can be deployed in standard RL settings. We first sketch how we modeled neurons and synapses to build up the TW circuit.

Neuron & Synapse Model – Most of the neurons in *C. elegans* are observed to exhibit electrotonic dynamics [Kato *et al.*, 2015], meaning that electric charges spread passively inside a neuron creating graded potentials. This implies that the neurons are non-spiking. Dynamics of the neurons' membrane potential therefore, were modeled by the well-known, deterministic ordinary differential equation (ODE), the *single-compartment membrane equation* [Koch and Segev, 1998]: $C_m \frac{dv_i}{dt} = G_{Leak}\Big(V_{Leak} - v_i(t)\Big) + \sum_{i=1}^{n} I_{in}^{(i)}$, where C_m, G_{Leak} and V_{Leak} are parameters of the neuron and $I_{in}^{(i)}$, stands for the external currents to the cell. We adopted Eq. (1) to govern **interneurons**' dynamics.

For interacting with the environment, We adopted sensory and motor neuron models from [Lechner *et al.*, 2017]

Chemical synapses The chemical synaptic current depends on a non-linear component standing for their conductance strength, which is a function of the presynaptic neurons' potential, V_{pre}, and have a maximum weight of w,(standing for the maximum conductance of the synapse) as, [Koch and Segev, 1998]: $g(V_{pre}) = w/1 + e^{\sigma(V_{pre}+\mu)}$. Moreover, the synaptic current linearly depends on the postsynaptic neuron's membrane potential, V_{post}, and therefore can be formulated as, [Koch and Segev, 1998; Hasani *et al.*, 2017]: $I_s = g(V_{pre})(E - V_{post})$, where by varying E, the reversal potential of the synapse, it realizes inhibitory or excitatory connection to their postsynaptic neurons. An electrical synapse (**gap-junction**), was modeled by a constant conductance, $\hat{\omega}$, where based on the Ohm's law their bidirectional current between neurons j and i, can be computed as $\hat{I}_{i,j} = \hat{\omega}\Big(v_j(t) - v_i(t)\Big)$.

For simulating neural networks composed of such dynamic models, we adopted an implicit numerical solver [Press *et al.*, 2007]. Formally, we realized the ODE models in a hybrid fashion which combine both implicit and explicit Euler's method. See [Lechner *et al.*, 2018], for a concrete discussion on the model implementation, and the choice of parameters.

3 Search-based Reinforcement Learning

We formulated an RL setting for training the parameters of the neural circuit to perform the balancing of the inverted pendulum, control the mountain car, and to park the rover robot.

The behavior of a neural circuit can be expressed as a policy $\pi_\theta(o_i, s_i) \mapsto \langle a_{i+1}, s_{i+1}\rangle$, that maps an observation o_i, and an internal state s_i of the circuit, to an action a_{i+1}, and a new internal state s_{i+1}. This policy acts upon a possible stochastic environment $Env(a_{i+1})$, that provides an observation o_{i+1}, and a reward, r_{i+1}. The stochastic return is given by $R(\theta) := \sum_{t=1}^{T} r_t$.

Objective of the *Reinforcement learning* is to find a θ that maximizes $\mathbb{E}\big(R(\theta)\big)$.

Search-based methods directly randomly sample parameters and estimate how good these random parameters are, to update

Table 1: Mapping the environmental variables to the sensory and motor neurons of the TW circuit, in different experiments

Experiment	*Environment variable*	*Type*	*Positive neuron*	*Negative neuron*
Inverted Pendulum	φ	Sensor (pendulum angle)	PLM	AVM
	x	Sensor (cart position)	ALM	PVD
	a (Control)	Motor (move right/left)	FWD	REV
Mountain Car (OpenAI Gym)	x	Sensor (car position)	PLM	AVM
	$\dot{x}$	Sensor (car's linear velocity)	ALM	PVD
	a (Control)	Motor (move right/left)	FWD	REV
Mountain Car (rllab)	x	Sensor (car position)	PLM	AVM
	$\dot{x}$	Sensor (car's linear velocity)	ALM	PVD
	a (Control)	Motor (move right/left)	FWD	REV
Cart-Pole	φ	Sensor (pole angle)	PLM	AVM
	$\dot{\varphi}$	Sensor (pole angular velocity)	ALM	PVD
	a (Control)	Motor (move right/left)	FWD	REV
Parking of a Rover	x	Sensor (estimated x position)	PVD	
	y	Sensor (estimated y position)	PLM	
	s	Sensor(start signal)	AVM	
	θ	Sensor (estimated angular pose)	ALM	
	a_1 (Control)	Motor (angular velocity)	FWD	REV
	a_2 (Control)	Motor (linear velocity)	FWD/REV	

θ [Salimans *et al.*, 2017; Szita and Lörincz, 2006]. Here, we adopted such search-based optimization which can be applied in any control setting, regardless of the internal structure of the policy, (*black-box optimization*). Our approach is based on a *Random Search* (RS) [Rastrigin, 1963] optimization, combined with an *Objective Estimate* (OE) as an objective function $f : \theta \mapsto \mathbb{R}^+$.

The OE generates N rollouts with π_θ on the environment and computes an estimate of $\mathbb{E}(R_\theta)$ based on a filtering mechanism on these N samples. To ensure that a single outlying high OE for some θ does not hinder the algorithm to find a legitimately good parameter, the OE is reevaluated after it is utilized M times, within the algorithm. The full algorithm is outlined in Algorithm 1.

4 Experiments

We prepared five environmental settings for the TW sensory/motor neurons and then deployed our RL algorithm to learn the parameters of the TW circuit to realize the given control objective. Environments include I) Inverted pendulum of Roboschool [Schulman *et al.*, 2017], II) Mountain car of OpenAI Gym, III) Mountain car of rllab, IV) cart-pole balancing of rllab and V) Parking a real rover robot from a learned policy in a simulated environment. The code is available online [1].

Inverted pendulum– The TW neural circuit shown in Figure 1, contains four sensory neurons. It therefore, allows us to map the circuit to only two input variables. Table 1 summarizes the mapping of the environmental variables to the tap-withdrawal circuit. We set up the search-based RL algorithm to optimize the neurons and synapses parameters $\omega, \hat{\omega}, \sigma, C_m, V_{Leak}$ and G_{Leak} in the Roboschool `RoboschoolInvertedPendulum-v1` environment with a slight modification in the reward calculation [Schulman *et al.*, 2017]. A video of different stages of the learned neuronal circuit policy for the inverted pendulum can be viewed at `https://youtu.be/iOHeQ7DhQv8`.

Similar to the Inverted pendulum problem, we mapped the environmental variables of the **Mountain-car** and the **cartepole** to the TW circuit based on Table 1 and learned their parameters. A video illustrating the control of the car at various episodes during the learning process of the mountain car problem, can be viewed at `https://youtu.be/lMrP1sXp3jk`.

Autonomous parking of the rover– In this experiment, we generalized our TW neuronal circuit policy to a real-world control setting. We let the TW circuit learn to park a rover robot on a determined spot, given a set of checkpoints which form a trajectory, in a deterministic simulated environment. We then deployed the learned policy on a mobile robot in a real environment. The mapping details are provided in Table 1. Accordingly, the TW circuit is able to govern the robot's locomotion in three different directions, forward, left- and right-turns.

We then learned the TW circuit, by the RL algorithm. The learned policy has been mounted on a Pioneer AT-3 mobile robot and performed a reasonable parking performance. The Video of the performance of the TW neuronal circuit policy on the parking task can be viewed at `https://youtu.be/Vwydc2ez9Wc`.

4.1 Performance

The training algorithm was able to solve all five tasks, after a reasonable number of iterations as shown in Figure 2. All learning curves reached a stable state after the given number of iterations. The final return values for the basic standard RL tasks (provided in Table 2), matches that of conventional policies [Heidrich-Meisner and Igel, 2008], and the state-of-the-art deep neural network policies learned by many RL algorithms [de Froissard de Broissia and Sigaud, 2016; Schulman *et al.*, 2017; Berkenkamp *et al.*, 2017].

Table 2: Training results. Return value $\pm$ standard deviation. Performance of the learned neuronal circuit policies in terms of final return and average return over all training iterations.

Environment	**Final return**	**Average return**
Inverted pendulum	1168.5 ± 21.7	394.1 ± 80.9
Mountain car (Gym)	91.5 ± 6.6	28.1 ± 9.6
Mountain car (rllab)	-61.3 ± 1.4	-196.3 ± 78.0
Cart-pole balancing	3716.1 ± 240.1	1230.3 ± 272.2
Parking	-0.49 ± 0.63	-6.13 ± 1.41

[1] Code for all experiments is available online at: `https://github.com/mlech26l/neuronal_circuit_policies`

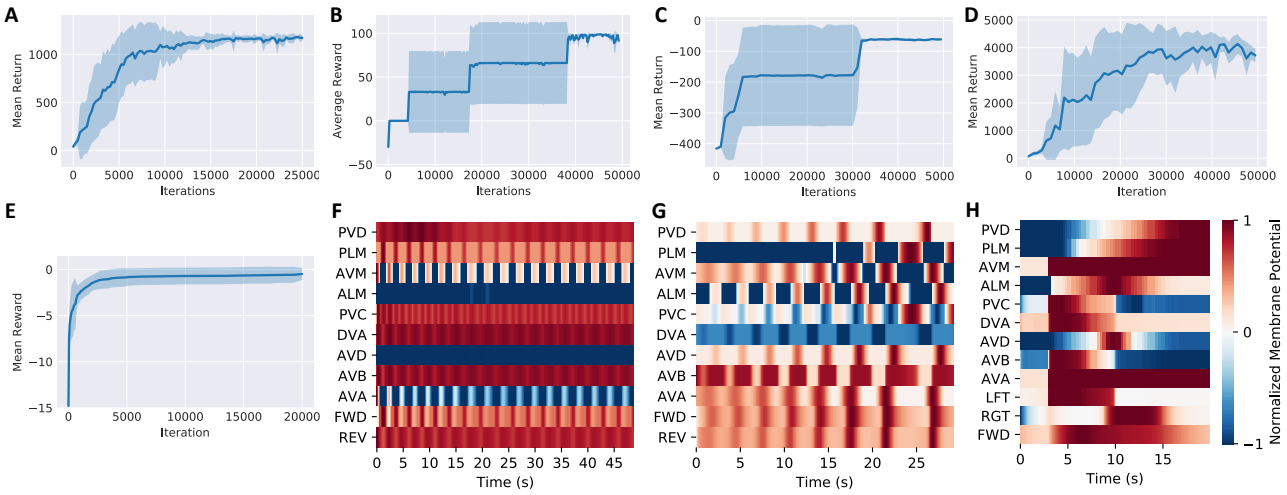


Figure 2: Performance as a function of the number of iterations. A) Inverted pendulum B) Mountain car (OpenAI Gym) C) Mountain car (rllab) D) Cart-pole (rllab) E) parking trajectory of the rover robot F) Influence of filter size on training performance. F-G) Neuronal activity of the Learned neuronal policies for the inverted pendulum, the mountain car (OpenAI Gym) and the parking.

4.2 Interpretability of the neuronal circuit policies

Interpretability of neural network policies is still a challenge to be solved [Heess *et al.*, 2016; Bacon *et al.*, 2017]. Successful attempts on the development of interpretable representation learning have been provided where the latent space of the learned policies demonstrate interpretable skills [Chen *et al.*, 2016; Florensa *et al.*, 2017]. A neuronal circuit policy has the distinct attribute of being interpretable at the cell level.

Inverted Pendulum – Activity of the learned Tap-withdrawal's neurons in a successful episode of the inverted pendulum control, is shown in Figure 2F. The learned circuit surprisingly realized a competitive behavior between two sub-circuits within the network, similar to the reflexive behavior observed in the worm [Wicks *et al.*, 1996]. Neurons AVM, AVD, AVA, and REV control the pendulum not to fall on the left side, whereas a circuit composed of PLM, PVC, AVB and FWD, controls the other side of the balance. The antagonistic behavior between the command neurons AVA and AVB balances the pendulum in the middle. Note that DVA neuron acts as a mediator which couples the kinetics of the two sub-circuits. Note that DVA neuron acts as a mediator which couples the kinetics of the two sub-circuits. Note that Neurons' dynamics are realized with a variety of time-constants during the balancing control episodes. For instance PVC modulates fast undulations while the motor neurons work with a slower dynamic.

Mountain Car – As illustrated in Figure 2G, a gradual increase in the amplitude of the overall periodic dynamic of the neurons is observed due to the alternation of the direction of movement to bring the car uphill. Sensory neurons linearly transform the environmental observations to the global activity of the system, forming a phase-shifted highly synchronized behavior. In every episode, by the rising of PDV (higher negative velocity) or PLM (be at a positive x), all neurons approach their resting potential. This feedback mechanism which prevents unstable behavior, originated from inhibition of PVC and AVD by the ALM sensory neuron. At every period in which the car reaches the maximum position on the left-hand side, a circuit composed of AVM, AVD, AVB and FWD applies a force-pulse which pushes the car to get closer to the target. AVA and AVB retained their objective as in the TW actual circuit, as the command neurons which enforce the network's decision to the motor neurons REV and FWF. However their activity is not fully antagonistic in this learned network. This is where the competition "A more active command neuron, gives rise to a more active downstream motor neuron, and as a results winning the competition", is still present.

Parking of the rover - Figure 2H shows the activity of the learned TW policy on the parking task. AVM receives the starting signal and accordingly, a left turn together with moving forward are initiated by the motor neuron FWD and LFT. This is governed by the command neuron AVB. The rover continues moving forward and initializes a right turn by the motor neurons FWD and RGT. This is controlled by AVA. AVD learned to control the turning-phases of the rover by moderating its state, (a higher potential during right turns and a lower membrane state during left turns). PVC and DVA function as the pre-command neurons which incorporate the sensory and the network inputs, into a readable action for the command neuron AVB.

5 Conclusions

We showed how neuronal circuit policies can realize interpretable dynamics in standard control and RL tasks. Our neuronal policies are limited to the domains in which the sensory observations and motor actions are as many as the available sensory and motor neurons of the neural circuit. Implementation of larger neural circuits to build controllers for more complex tasks will be the focus of our continued effort. Moreover, a policy gradient algorithm may significantly enhance the performance of our policies.

Acknowledgements

Authors would like to thank Jean V. De Carvalho for providing constructive feedbacks on the manuscript. This work was supported with computation resources by Microsoft Azure via the Microsoft Azure for Research Award program.

References

[Ardiel and Rankin, 2010] Evan L Ardiel and Catharine H Rankin. An elegant mind: learning and memory in caenorhabditis elegans. *Learning & memory*, 17(4):191–201, 2010.

[Bacon *et al.*, 2017] Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In *AAAI*, pages 1726–1734, 2017.

[Bargmann, 2006] Cornelia I Bargmann. Chemosensation in c. elegans. *WormBook*, pages 1–29, 2006.

[Berkenkamp *et al.*, 2017] Felix Berkenkamp, Matteo Turchetta, Angela Schoellig, and Andreas Krause. Safe model-based reinforcement learning with stability guarantees. In *Advances in Neural Information Processing Systems 30*, pages 908–919. Curran Associates, Inc., 2017.

[Chalfie *et al.*, 1985a] M Chalfie, JE Sulston, JG White, E Southgate, JN Thomson, and S Brenner. The neural circuit for touch sensitivity in *Caenorhabditis elegans*. *Journal of Neuroscience*, 5(4):956–964, 1985.

[Chalfie *et al.*, 1985b] Martin Chalfie, John E Sulston, JOHN G White, Eileen Southgate, J Nicol Thomson, and Sydney Brenner. The neural circuit for touch sensitivity in caenorhabditis elegans. *Journal of Neuroscience*, 5(4):956–964, 1985.

[Chen *et al.*, 2006] Beth L. Chen, David H. Hall, and Dmitri B. Chklovskii. Wiring optimization can relate neuronal structure and function. *Proceedings of the National Academy of Sciences of the United States of America*, 103(12):4723–4728, 2006.

[Chen *et al.*, 2016] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2172–2180, 2016.

[Culotti and Russell, 1978] Joseph G Culotti and Richard L Russell. Osmotic avoidance defective mutants of the nematode caenorhabditis elegans. *Genetics*, 90(2):243–256, 1978.

[de Froissard de Broissia and Sigaud, 2016] Arnaud de Froissard de Broissia and Olivier Sigaud. Actor-critic versus direct policy search: a comparison based on sample complexity. *CoRR*, abs/1606.09152, 2016.

[Doya, 2000] Kenji Doya. Reinforcement learning in continuous time and space. *Neural computation*, 12(1):219–245, 2000.

[Duan *et al.*, 2016] Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking deep reinforcement learning for continuous control. In *International Conference on Machine Learning*, pages 1329–1338, 2016.

[Florensa *et al.*, 2017] Carlos Florensa, Yan Duan, and Pieter Abbeel. Stochastic neural networks for hierarchical reinforcement learning. *arXiv preprint arXiv:1704.03012*, 2017.

[Hasani *et al.*, 2017] Ramin M Hasani, Victoria Beneder, Magdalena Fuchs, David Lung, and Radu Grosu. Sim-ce: An advanced simulink platform for studying the brain of caenorhabditis elegans. *arXiv preprint arXiv:1703.06270*, 2017.

[Heess *et al.*, 2016] Nicolas Heess, Greg Wayne, Yuval Tassa, Timothy Lillicrap, Martin Riedmiller, and David Silver. Learning and transfer of modulated locomotor controllers. *arXiv preprint arXiv:1610.05182*, 2016.

[Heidrich-Meisner and Igel, 2008] Verena Heidrich-Meisner and Christian Igel. Variable metric reinforcement learning methods applied to the noisy mountain car problem. In *European Workshop on Reinforcement Learning*, pages 136–150. Springer, 2008.

[Kato *et al.*, 2015] Saul Kato, Harris S. Kaplan, Tina Schrödel, Susanne Skora, Theodore H. Lindsay, Eviatar Yemini, Shawn Lockery, and Manuel Zimmer. Global brain dynamics embed the motor command sequence of *Caenorhabditis elegans*. *Cell*, 163:656–669, October 2015.

[Koch and Segev, 1998] Christof Koch and Koch Segev. *Methods in Neuronal Modeling - From Ions to Networks*. MIT press, second edition, 6 1998.

[Lechner *et al.*, 2017] Mathias Lechner, Radu Grosu, and Ramin M Hasani. Worm-level control through search-based reinforcement *arXiv preprint arXiv:1711.03467*, 2017.

[Lechner *et al.*, 2018] Mathias Lechner, Ramin M Hasani, and Radu Grosu. Neuronal circuit policies. *arXiv preprint arXiv:1803.08554*, 2018.

[Li *et al.*, 2012] Zhaoyu Li, Yidong Li, Yalan Yi, Wenming Huang, Song Yang, Weipin Niu, Li Zhang, Zijing Xu, Anlian Qu, Zhengxing Wu, and Tao Xu. Dissecting a central flip-flop circuit that integrates contradictory sensory cues in *C. elegans* feeding regulation. 3:776 EP –, Apr 2012.

[Moore, 1990] Andrew William Moore. Efficient memory-based learning for robot control. 1990.

[Nichols *et al.*, 2017] Annika LA Nichols, Tomáš Eichler, Richard Latham, and Manuel Zimmer. A global brain state underlies c. elegans sleep behavior. *Science*, 356(6344):eaam6851, 2017.

[Press *et al.*, 2007] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. Cambridge University Press, New York, NY, USA, 3 edition, 2007.

[Rankin *et al.*, 1990] Catherine H Rankin, Christine DO Beck, and Catherine M Chiba. Caenorhabditis elegans: a new model system for the study of learning and memory. *Behavioural brain research*, 37(1):89–92, 1990.

[Rastrigin, 1963] L. A. Rastrigin. About convergence of random search method in extremal control of multi-parameter systems. *Avtomat. i Telemekh.*, 24:1467–1473, 1963.

[Russell and Norvig, 2010] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, 3 edition, 2010.

[Salimans *et al.*, 2017] Tim Salimans, Jonathan Ho, Xi Chen, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. 2017.

[Schulman *et al.*, 2017] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[Singh and Sutton, 1996] Satinder P Singh and Richard S Sutton. Reinforcement learning with replacing eligibility traces. *Recent Advances in Reinforcement Learning*, pages 123–158, 1996.

[Szita and Lörincz, 2006] I. Szita and A. Lörincz. Learning tetris using the noisy cross-entropy method. *Neural Computation*, 18(12):2936–2941, 2006.

[Wen *et al.*, 2012] Quan Wen, Michelle D Po, Elizabeth Hulme, Sway Chen, Xinyu Liu, Sen Wai Kwok, Marc Gershow, Andrew M Leifer, Victoria Butler, Christopher Fang-Yen, et al. Proprioceptive coupling within motor neurons drives c. elegans forward locomotion. *Neuron*, 76(4):750–761, 2012.

[White *et al.*, 1986] J. G. White, E. Southgate, J. N. Thomson, and S. Brenner. The structure of the nervous system of the nematode *Caenorhabditis elegans*. *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, 314(1165):1–340, 1986.

[Wicks and Rankin, 1995] SR Wicks and CH Rankin. Integration of mechanosensory stimuli in *Caenorhabditis elegans*. *Journal of Neuroscience*, 15(3):2434–2444, 1995.

[Wicks *et al.*, 1996] Stephen R. Wicks, Chris J. Roehrig, and Catharine H. Rankin. A dynamic network simulation of the nematode tap withdrawal circuit: Predictions concerning synaptic function using behavioral criteria. *Journal of Neuroscience*, 16(12):4017–4031, 1996.

[Widrow, 1964] Bernard Widrow. Pattern recognition and adaptive control. *IEEE Transactions on Applications and Industry*, 83(74):269–277, 1964.

Exploring Explainable Artificial Intelligence and Autonomy through Provenance

Crisrael Lucero, Braulio Coronado, Oliver Hui, Douglas S. Lange
Space and Naval Warfare Systems Center Pacific
{crisrael.lucero, braulio.coronado, oliver.hui, doug.lange}@navy.mil

Abstract

With the issues of trust and transparency within artificial intelligence, it is difficult to place the decision making process in the hands of possibly biased or erroneous algorithms. Provenance is defined as a record that describes the people, institutions, entities, and activities involved in producing, influencing, or delivering a piece of data or a thing [Moreau and Groth, 2013]. In this paper, we seek to outline the process of using data provenance as a medium to accomplish explainability in AI and autonomy.

1 Introduction

A key component of an articial intelligence (AI) system is the ability to explain the decisions, recommendations, predictions, or actions made by it and the process through which they are made. Systems are interpretable if their operations can be understood by a human, either through introspection or a produced explanation [Biran and Cotton, 2017]. Due to the increased usage of routine machine learning algorithms on user data, policymakers are fighting to give users a "right to explanation" from algorithmic decisions that were made about them [Goodman and Flaxman, 2016]. Some automated decisions being made could significantly affect the user and potentially cause harm.

[ACM, 2017] and [Thelisson *et al.*, 2017] state that computational models can be distorted as a result of biases contained in their input data and/or their algorithms. This further propagates the idea that decisions and plans being made by AI should be explained to a human before being executed. From the ACM U.S. Public Policy Council statement, data provenance is an explicit principle for algorithmic transparency and accountability. Several other principles have strong relations to provenance such as accountability and auditability. A proposed solution for regulatory mechanisms was to increase the transparency in the data chain [Thelisson *et al.*, 2017]. This solution focuses on data controllers providing a satisfactory explanation for specific automated decisions by giving reasons to AI/ML actions. Provenance seeks to answer this by providing the process flow and change in data as a system continues to evolve.

Provenance was originally the chronology of the ownership, custody, or location of a historical object. A provenance record was kept with works of art to prove accountability, authenticity, and trust for vendors and traders. The PROV-DM is a generic data model for provenance that allows domain and specific representations of provenance to be translated into such a data model and interchanged between systems [Moreau *et al.*, 2013].

This paper explores how we approach a provenance-based methodology to practice explainable artificial intelligence in the systems we're targeting in order to accomplish this.

2 Background and Related Work

Research efforts involving provenance have made significant contributions towards data mining and extracting knowledge from provenance graphs. Statistical reasoning has been used to determine whether users and associated data were trustworthy within a crowdsourcing, provenance-enabled application [Keshavarz *et al.*, 2015]. Keshavarz's work defined a generic application-independent approach to interpret provenance data to make online decisions. Other assessments have been made such as a using machine learning techniques to build predictive models on the trustworthiness of data from users [Huynh *et al.*, 2013]. Furthermore, analysis of the provenance data can lead to behavior recognition from the crowd's activities.

Other directions that have been taken describes decisions as the provenance of the selected course of action [Putnam *et al.*, 2017]. Decisions are made by taking into consideration the information fusion of constraints, goals, dependencies, and the current scenario. The W3C PROV standard can be utilized, along with specific domain vocabulary, to enable this decision representation. Putnam suggests that a decision format using provenance could be used as a machine-understandable representation of the history of ideas. An explanation can be derived from the history of data for decisions made. [Moreau *et al.*, 2013] already states the interoperability of exchanging provenance between systems, but expanding upon this would allow for increased human-computer interaction. Provenance of human actions and decisions should be shared with machines, similar to [Huynh *et al.*, 2013] but should also be shared back to humans to provide explanations of the system of systems.

Similarly, a study has been done to create models of explainability from controlling robots by having them justify their actions [Sheh, 2017]. In this work, a robot traversing

through terrain would give a reason they chose to do a certain action based on learned traversal. The robot essentially explains how they got to their desired action based on the goal and the learned decision tree. In our case study, we go over an incident-based explanation using provenance that provides activities and changes in data that lead autonomous software systems to perform certain actions, similar to the robot listing reasons based on decision tree nodes.

3 Methodology

One of the techniques used in [Brinton, 2017] was the "Gray-Box Decision Characterization" (GBDC) approach. The GBDC technique probes the output of a machine learning model along the dimensions of the explainable framework. GBDC has some knowledge of the inner-workings of the machine learning algorithm, but does not change any aspect of it. The objective of this technique is to provide an explanation for single, specific outputs.

Our methodology is very similar in a way where a provenance software service isn't necessarily embedded into the autonomous systems, but rather observes the actions and instruments key agents and entities. The approach is "gray-box"-like because the actions that an autonomous system can make must be known beforehand. When an autonomous actor does anything, provenance is generated and models the actions done by them. This is accomplished using a provenance templating system [Michaelides *et al.*, 2014] to generate PROV. With PROV-Templates, we create predefined templates that can describe the various actions a system can accomplish.

During system runtime, the provenance service will create bindings that model the system and merge the new provenance with currently existing provenance graphs. The bindings, which are associations between the system data and template variables, are used to create instantiations of the predefined templates and generate provenance. The generated provenance can then be parsed, queried, and analyzed for explainability and data mining.

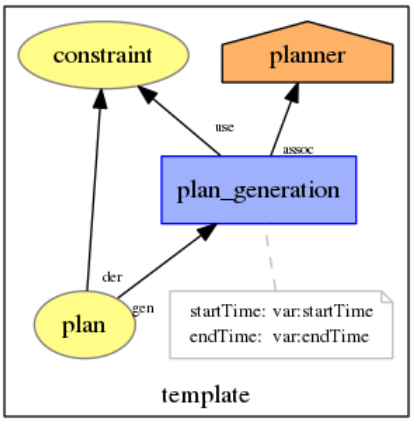


Figure 1: The template defines a generic action, entities, and an agent. A binding with data is used to instantiate the template and generate provenance.

The traditional PROV layout convention represents entities as yellow ellipses, activities as blue rectangles, and agents as orange pentagon-houses [Moreau and Groth, 2013]. PROV-DM offers a plethora of relationships between the different social constructs (agents, entities, and activities). Some of these relationships demonstrated in the figure are associations between an activity and agent, derivation between entities, generation between an entity and activity, and an activity using an entity. Variables and tags can be added upon each social construct such as the startTime and endTime on the activity as seen above, roles can be added to agents, and customized information can be attached to any of them.

3.1 Case Study: IMPACT

The Intelligent Multi-UxV Planner with Adaptive Collaborative Control Technologies (IMPACT) is a prototype Command and Control (C2) system. The IMPACT system practices centralized supervisory control of several autonomous unmanned vehicles. Both high level C2 planning and platform level vehicle autonomy are practiced to grant a single operator control of teams of disparate unmanned vehicles [Gutzwiller *et al.*, 2015]. An IMPACT operator uses a "playbook" approach to mission tasking by determining the type of mission and constraints required to complete a task. The built in application-level autonomy will allocate the appropriate resources and unmanned vehicles, and the vehicles have built-in, platform-level autonomy to accomplish the "play" generated by IMPACT.

Several software services exist within or alongside IMPACT such as the Plan Monitor, the Fusion agent, and an autonomous Task Management agent. These services act as autonomous agents that practice autonomic computing, machine learning, and formal methods to accomplish C2 tasking [Coronado *et al.*, 2016]. The Plan Monitor uses the Rainbow framework for analyzing mission health, and is further explained in a later section. The Task Manager contains a built-in autonomous agent that the operator can assign for it to complete at its own discretion.

In the current state of provenance being used in IMPACT, we provide information with regards to the actors and activities that lead up to a certain action. Using the Gray-Box Provenance Modeling method, our provenance instrumentation service subscribes to a centralized messaging Hub within IMPACT and treat the entire system, alongside all the services that communicate within, as a "gray-box". We take the above template and create a binding with JSON that contains information on the system state and actions of several services within the system. We expand and instantiate the template with that binding to create Figure 2. The gray-box approach makes it so that a template for every important action and decision is required to be made, so internal knowledge of the inner machinations is needed for the design and implementation.

An expansion of the template when filled with information from a binding shows that several instantiated entities can be formed from a single template entity. From this excerpt of a single piece of provenance, we can see that the planner, Fusion, generated a Plan based on certain constraints provided (mission type, weather conditions, and sensors re-

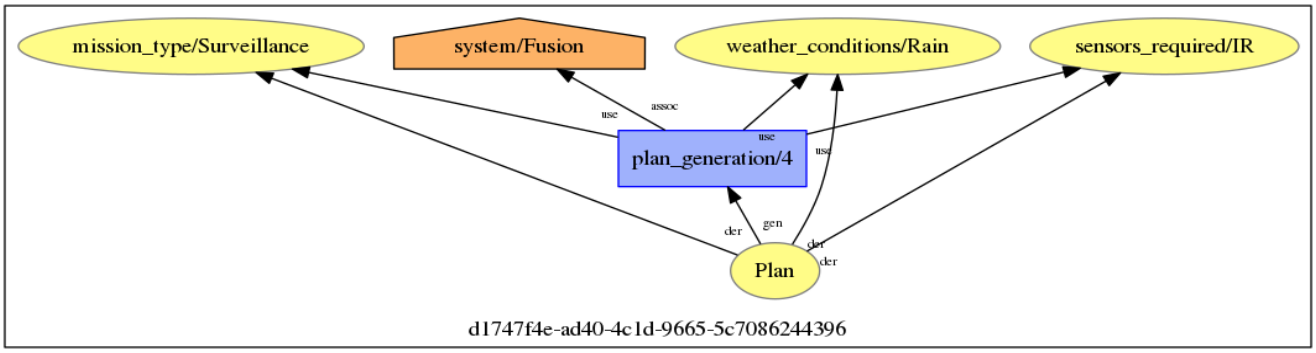


Figure 2: An example of the template from Figure 1 being instantiated with data from a binding.

quired). Within the IMPACT system, there are moments when autonomous agents must take over to correct an issue. If other systems introduce pop-up no fly zones or an unmanned vehicle is leaving the communications range, action is necessary immediately. If an operator is inattentive and slow to respond, autonomy will take over and complete the task. It is essential to relay this information to the operator; querying the generated provenance will allow software services to discover which autonomous agents made decisions, what assets and vehicles were involved, and why they needed to make those actions (i.e. Plan Monitor detects a pop-up no fly zone and re-routes an unmanned aerial vehicle that was originally on an over-watch mission). Further tracing and querying through the provenance can help extract even more information.

3.2 Case Study: Rainbow

The Plan Monitor service that exists within IMPACT utilizes the Rainbow framework for self-adaptation and autonomic computing [Coronado *et al.*, 2016]. Rainbow is a framework developed by Carnegie Mellon University for developing self-adaptive systems that focus on architectural models of systems to monitor, reason about, and adapt [Garland *et al.*, 2004]. We investigated the usage of instrumenting the Rainbow framework and modeling the activities in the PROV-DM. Applying the Gray-Box Provenance Modeling method is straightforward due to the Rainbow framework having built-in instrumentation and logging.

Templates would be created to observe Rainbow Probes as they extract data from a system and how a Gauge reacts and updates the system model based on data that Probes have extracted. If the system model does not comply with a predefined rule, which is noticed by the Architecture Evaluator, it triggers a violation. The Adaptation Manager will then try to address this violation by selecting a predefined strategy and the Strategy Executor will carry it out. This will put the system model into a state to be probed again.

Rainbow uses predefined rules and strategies for self-adaptation and autonomic computing, but modeling the framework in provenance could shed light into deriving what initially unintuitive changes in a system is causing certain rules to be violated. For example, within the IMPACT system red units may appear and an autonomous agent can be tasked with handling the situation. The red unit was outside of communications range, so Rainbow within Plan Monitor will realize the change in a specific system state (i.e. not all units are within communications range), cause a rule violation to trigger, and execute an appropriate strategy (i.e. perform a communications relay with another unit). From an autonomics standpoint, this is straight forward as the system is able to self-heal and self-adapt [Murch, 2004]. Introducing provenance allows potential relationships to be revealed. From the previous example, assume that we can clearly see several communications relays are being issued by the Plan Monitor. Querying the provenance graphs may reveal that the type of vehicles being used by the red team were unmanned, commodity vehicles with no threat capability and the locations the tracks were strategically placed in a way that spreads our forces thin in a critical area of interest.

4 An Example for Provenance for AI

Although we showcased the usage of PROV for autonomous software agents in an adaptive system, we introduce a generic way of modeling an AI algorithm based on the outputs and processing of data. As previously stated, as the algorithm is processing data, important steps and actions should be recorded in order to be modeled in the PROV-DM. Templates should be designed based on the different activities an algorithm performs due to inputted data.

Figure 3 showcases how a very basic data flow for a machine learning algorithm could be tracked, recorded, and modeled in provenance. Several of the entities that are being derived and generated are members of multiple templates, not completely expressed in the figure. The instantiated templates form bundles, which merge intelligently based on the unique IDs of the entities. An example are the ‘feature’ entities which are members of bundle 2 and 3; although they only show up as members of bundle 2, the templates and bindings account for them being a part of bundle 3, as well.

In practice, significantly more than this should be modeled, and labels with useful information should be present. Whether it be datasets, features, or outputs, a plethora of information should be being modeled into provenance. This can increase the accountability and transparency of which data is being fed into the algorithm and if there are notable

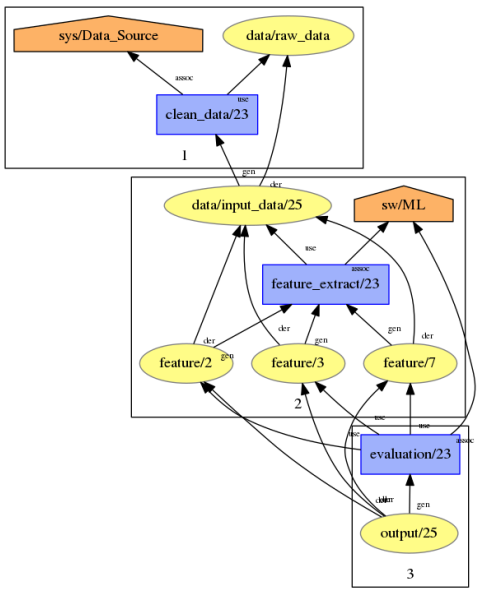


Figure 3: An example of how some explainability can be derived from AI and machine learning.

features we can discover. Provenance is already being used for recording scientific workflow in order to increase the trust and validity of experiments and results [Davidson *et al.*, 2007]. If AI/ML algorithms should be held under the same standards as human decision makers [Thelisson *et al.*, 2017; Goodman and Flaxman, 2016], then provenance for AI is necessary.

5 Explainability and Accountability

We intend to explore different avenues of explanation using provenance within the IMPACT system. The current state provides incident-based explanations whenever autonomous software agents make decisions without the human-in-the-loop and provide information as to what triggered the actions to be made. The current approach is within its infancy and our future work seeks to expand upon it to provide more explainability.

Our current direction is exploring different data mining and knowledge-extraction techniques that have been developed while also developing new, novel methods. One such technique is Provenance Summarization [Moreau and Groth, 2015], which seeks to summarize the graph data and offer explanations based on the structure. As provenance graphs grow due to many actions occurring within a system, the overwhelming amount of data can no longer be comprehended by a human. The summarization technique allows for extremely large provenance graphs to be visualized in a way that is understandable and not overwhelming. Summarization is also a way to resolve scalability issues. A common problem when instrumenting an algorithm or system with provenance is determining level of granularity that must be established. Some studies approach the problem with active transformers and lenses that analyze specific aspects of the graph to keep up with scalability issues, such as [Gehani *et al.*, 2016]. Although this excludes and abstracts data that isn't necessarily needed by the client, one could argue that you lose some accountability when not modeling every aspect of a system or algorithm. Each problem is unique, and requires well-tailored provenance design to determine what is necessary and what level of a system should be observed. Too high of a granularity level and you have an overabundance of potentially useless data; yet, too low of a granularity may not derive any interesting data.

Not only does provenance allow us to provide a form of explanation, it is a critical piece in achieving accountability as well. [Kirkpatrick, 2016] states that regulatory oversight is needed given that algorithms, and the way they are used, can do significant harm to many people. The study showed that metrics between the AI that tried to correlate potential crimes with current ones held heavy bias in certain characteristics and traits. Using provenance, a light can be shed on these biases, as a record trail can show exactly what influences lead to an action or decision made by AI.

6 Conclusion

In this study, we explored an avenue of using provenance for explainable artificial intelligence and autonomy. There's an increasing need to build accountability and explanations for the actions of AI algorithms. The natural properties described by provenance in the world of art can translate to the AI realm. We discussed a methodology for implementing the PROV-DM in a gray-box-like framework using provenance templates.

Although our work in using provenance for XAI is still in its infancy, there are several cases of provenance for explainability in other fields to aid our venture. One such area of future work is analyzing provenance at runtime to provide sophisticated incident-based and scenario-based explanations. In the previously stated situation where the red teams spread the units using commodity unmanned vehicles, there should be responsive incident-based capabilities notifying operators of potential threats based on this relationship discovered through provenance. Scenario-based capabilities can explain, possibly through the usage of a summarization technique, an abstracted explanation.

References

[ACM, 2017] US ACM. Statement on algorithmic transparency and accountability. *USACM Press Releases*, 9(2):1–2, January 2017.

[Biran and Cotton, 2017] Or Biran and Courtenay Cotton. Explanation and justification in machine learning: A survey. In *IJCAI 2017 Workshop on Explainable Artificial Intelligence*, 2017.

[Brinton, 2017] Chris Brinton. A framework for explanation of machine learning decisions. In *IJCAI 2017 Workshop on Explainable Artificial Intelligence*, 2017.

[Coronado *et al.*, 2016] Braulio Coronado, Eric Gustafson, John Reeder, and Douglas S. Lange. Mixing formal methods, machine learning, and human interaction through an

autonomics framework. In *Proceedings of the 2016 AAAI Fall Symposium Series*, 2016.

[Davidson *et al.*, 2007] Susan Davidson, Sarah Cohen-Boulakia, Anat Eyal, Bertram Ludascher, Timothy McPhillips, Shawn Bowers, Manish Kumar Anand, and Juliana Freire. Provenance in scientific workflow systems. In *IEEE Data Bulletin Engineering*, 2007.

[Garland *et al.*, 2004] David Garland, Shang-Wen Cheng, An-Cheng Huang, Bradley Schmerl, and Peter Steenkiste. Rainbow: Architecture-based self adaptation with reusable infrastructure. *IEEE Computer*, 37(10), 2004.

[Gehani *et al.*, 2016] Ashish Gehani, Hasanat Kazmi, and Hassaan Irshad. Scaling spade to 'big provenance'. In *8th USENIX Workshop on the Theory and Practice of Provenance*, 2016.

[Goodman and Flaxman, 2016] Bryce Goodman and Seth Flaxman. European union regulations on algorithmic decision-making and a 'right to explain'. arXiv:1606.08813, 2016.

[Gutzwiller *et al.*, 2015] Robert S. Gutzwiller, Douglas S. Lange, John Reeder, Rob L. MOrris, and Olinda Rodas. Human-computer collaboratoin in adaptive supervisory control and function allocation. In *7th International Conference on Virtual, Augmented, and Mixed Reality*, pages 447–456. Springer, 2015.

[Huynh *et al.*, 2013] Trung Dong Huynh, Mark Ebden, Matteo Venanzi, Sarvapali D. Ramchurn, Stephen Roberts, and Luc Moreau. Interpretations of crowdsourced activities using provenance network analysis. In *Proceedings of the First AAAI Conference on Human Computation and Crowdsourcing*, pages 78–85, 2013.

[Keshavarz *et al.*, 2015] Amir Sezavar Keshavarz, Trung Dong Huynh, and Luc Moreau. Provenance for online decision making. In *Provenance and Annotation of Data and Processes*, 2015.

[Kirkpatrick, 2016] Keith Kirkpatrick. Battling algorithmic bias: How do we ensure algorithms treat us fairly? *Communications of the ACM*, 59(10):16–17, 2016.

[Michaelides *et al.*, 2014] Danius Michaelides, Trung Dong Huynh, and Luc Moreau. Prov-template: A template system for prov documents. `https://provenance.ecs.soton.ac.uk/prov-template`, 2014.

[Moreau and Groth, 2013] Luc Moreau and Paul Groth. *Provenance: An Introduction to PROV*. Morgan & Claypool, 2013.

[Moreau and Groth, 2015] Luc Moreau and Paul Groth. Aggregation by provenance types: A technique for summarising provenance graphs. In *Graphs as Models 2015 (ETAPS'15), Electronic Proceedings in Theoretical Computer Science*, pages 129–144, 2015.

[Moreau *et al.*, 2013] Luc Moreau, Paolo Missier, and eds. Prov-dm: The prov data model. `https://www.w3.org/TR/prov-dm`, 2013.

[Murch, 2004] Richard Murch. *Autonomic Computing*. IBM Press, 2004.

[Putnam *et al.*, 2017] Cheryl Putnam, Jeff Waters, and Olinda Rodas. A standard decision format using provenance. In *IEEE 17th International Symposium on Signal Processing and Information Technology*, 2017.

[Sheh, 2017] Raymond K. Sheh. 'why did you do that?' explainable intelligent robots. In *AAAI Workshop on Human-Aware Artificial Intelligence*, 2017.

[Thelisson *et al.*, 2017] Eva Thelisson, Kirtan Padh, and L. Elisa Celis. Regulatory mechanisms and algorithms towards trust in al/ml. In *IJCAI 2017 Workshop on Explainable Artificial Intelligence*, 2017.

Explore, Exploit, and Explain: Personalizing Explainable Recommendations with Bandits

James McInerney, Benjamin Lacker, Samantha Hansen, Karl Higley, Hugues Bouchard, Alois Gruson, Rishabh Mehrotra
Spotify
{ jamesm, benl, slhansen, karlhigley, hb, agruson, rishabhm }@spotify.com

Abstract

The multi-armed bandit is an important framework for balancing exploration with exploitation in recommendation. Exploitation recommends content (e.g., products, movies, music playlists) with the highest predicted user engagement and has traditionally been the focus of recommender systems. Exploration recommends content with uncertain predicted user engagement for the purpose of gathering more information. The importance of exploration has been recognized in recent years, particularly in settings with new users, new items, non-stationary preferences and attributes. In parallel, explaining recommendations ("recsplanations") is crucial if users are to understand their recommendations. Existing work has looked at bandits and explanations independently. We provide the first method that combines both in a principled manner. In particular, our method is able to jointly (1) learn which explanations each user responds to; (2) learn the best content to recommend for each user; and (3) balance exploration with exploitation to deal with uncertainty. Experiments with historical log data and tests with live production traffic in a large-scale music recommendation service show a significant improvement in user engagement.

1 Introduction

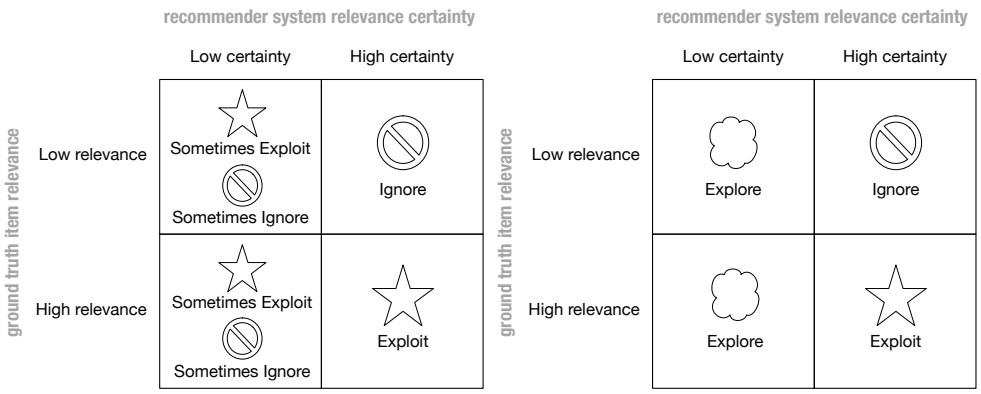
Recommender systems are central to the effectiveness of many online content and e-commerce platforms. Users are overwhelmed by the choice of what to watch, buy, read, and listen to online and recommendations are a popular way to help them navigate this choice. But recommendations without context lack motivation for a user to pay attention to them. Adding an associated explanation for a recommendation (abbreviated as *recsplanation* [Lamere, 2017]), is known to increase user satisfaction and the persuasiveness of recommendations [Kouki *et al.*, 2017; Friedrich and Zanker, 2011; Tintarev and Masthoff, 2007]

Explanations in recommender systems have the goals of both providing information about recommended items and encouraging better outcomes such as higher user engagement resulting from more confidence and transparency in the recommendations [Friedrich and Zanker, 2011]. In this paper, we argue that users respond to explanations differently and that, as a result, there is a need to jointly optimize both item selection and explanation selection for such systems. For example, more social users may respond better to explanations that reference activity in their social network than more private users. In addition, the same users may respond to explanations differently according to their context and intent, e.g., people on the move may require more dependable recommendations from their recent consumption history due to lack of time. We formalize the problem of jointly optimizing item and explanation recommendation and address the technical challenges this problem presents.

In more detail, introducing explanations to recommendation greatly expands the search space of actions the recommender can select from and compounds the problem of data sparsity. Recommendation methods that aim to maximize engagement without regard for model certainty or exploration can unnecessarily ignore highly relevant items. Figure 1a shows the various outcomes when a recommender can only exploit or ignore an item. When there is high certainty about the item relevance, resulting from an abundance of data about the user-item match, the recommender behaves optimally. However, in the face of uncertainty from only a small amount of data, the recommender will sometimes suboptimally ignore relevant items (the lower left quadrant in the grid). The fact that the recommender itself is active in deciding its own training data perpetuates this problem. The problem of ignoring items or whole categories of items leads to filter bubble pathologies [Chaney *et al.*, 2017].

Bandit approaches introduce the notion of exploration to reduce the uncertainty about the relevance of an item [Sutton and Barto, 1998]. By reserving judgement about item relevance until enough data has been collected, the bandit is able to discover more relevant items (the lower left quadrant of Figure 1b).

The aforementioned issues around item uncertainty also apply to explanation uncertainty. How to incorporate exploration when jointly optimizing for items and explanations is another gap in existing work. Naïvely treating each (*item*, *explanation*) pair as a distinct action would multiplicatively scale the number of actions that need to be explored. In addition, in some cases the recommender is required to provide multiple



(a) Exploitative methods have two modes: exploit or ignore an action.

(b) Bandit methods have three modes: exploit, ignore, or explore an action.

Figure 1: Recommendation methods that aim to maximize engagement without regard for model certainty or exploration sometimes unnecessarily ignore highly relevant items.

items within the same explanation (e.g., in the shelf layout on a website).

Our approach, BAndits for Recsplanations as Treatments (*Bart*), addresses these gaps in the following way. Bart learns and predicts satisfaction (e.g., click-through rate, consumption probability) for any combination of item, explanation, and context and, through careful logging and contextual bandit retraining, can learn from its mistakes in an online setting.

In summary, our contributions are the following:

- We identify and provide a formalization for the problem of jointly personalizing explainable recommendations.
- We present Bart, a contextual bandits-based framework that addresses the problem of recommending with explanations under uncertainty of user satisfaction.
- Implementing Bart in a scalable production environment presented challenges that we discuss and address.
- Through offline experiments on randomized historical data and online experiments with users in the homepage of a large-scale music recommender system, we empirically evaluate Bart and find that it significantly outperforms the best static ordering of explanations.

This paper is organized as follows. We consider work related to recommendations with explanations and bandits in Section 2. In Section 3, we formalize the problem and explain how Bart addresses it. We evaluate Bart in Section 4 using data offline and in a live production test. Finally, conclusions and future work are discussed in Section 5.

2 Related Work

Our work on personalizing explainable recommendations with bandits builds upon research in explaining recommendations and reinforcement learning for recommender systems. Explanations make recommendations more trustworthy and engaging [Kouki *et al.*, 2017]. They come in many different forms, from collaborative and social [Friedrich and Zanker, 2011] to visual [Kangasrääsiö *et al.*, 2015]. Contextual bandits have been in use with linear models for recommendation over the past several years [Li *et al.*, 2010]. Similar approaches have been applied to news recommendation [Shani *et al.*, 2005], travel information [Srivihok and Sukonmanee, 2005], and web pages [Joachims *et al.*, 1997]. Our work extends the state of the art with an approach that performs exploration-exploitation with very sparse contexts over the joint item-explanation space. Related work has also looked at slate recommendation for groups of items [Swaminathan *et al.*, 2017]. Our work is orthogonal to this research and could be adapted to work with slates.

3 Method

We next discuss the explainable recommendation problem in more detail in Section 3.1 and present Bart in Section 3.2.

3.1 Problem Formulation

The problem we address in this paper is how to jointly personalize recommendations of items with their associated explanation. Personalization is based on contextual features describing the user, their context, and item.

We formalize the problem in the following way. Let $\mathcal{J}$ be the set of items, $\mathcal{U}$ be the set of users, and $\mathcal{E}$ the set of explanations. Introduce the validity function $f : \mathcal{E} \times \mathcal{U} \rightarrow \mathcal{P}(\mathcal{J})$ that maps any pair of elements $e \in \mathcal{E}$ and $u \in \mathcal{U}$ to a subset of $\mathcal{J}$. In other words, f describes the set of valid items $V_{e,u} \subseteq \mathcal{J}$ for each explanation and user.

We limit our scope to assuming that f has been pre-specified by a designer. Note that this still admits a large number of possible explanations, through simple rules that are parameterized. Table 1 gives several examples of simple rules that can scalably cover the entire item set $\mathcal{J}$ and explanation set $\mathcal{E}$ through parameterization. Finally, note that since we have not restricted f to be injective, there may be more than one explanation for the same item.

Building on these definitions, the problem we address is that of building *a machine learning approach that sorts both*

Table 1: Examples of parameterized explanation rules

Parameterized Explanation	Rule for Generating Items	Example P's
Movies your friends love	Movies that are popular with P's friends	$\{ user_{123}, user_{921} \}$
More like P	Playlists by artists similar to P	{ David Bowie, Jay-Z, Lady Gaga }
Jump back in	Book genres that a user consumed heavily 6 months ago	$\emptyset$

Examples of simple parameterized rules for specifying the validity function f. Some of these rules assume side information that is commonly accessible, such as a way of finding similar artists through a graph or embedding representation.

explanations and items in a personal, dynamic, responsive, and relevant fashion. Formally, we seek a reward function $r : \mathcal{J} \times \mathcal{E} \times \mathcal{X} \rightarrow \mathbb{R}$ that accurately predicts the user engagement for item $j \in \mathcal{J}$, explanation $e \in \mathcal{E}$, given context $x \in \mathcal{X}$. The goal of training is to optimize the learning objective (which is related to the accuracy of the predicted reward) with respect to the free parameters of r.

In addition, given a reward function, we seek a method for selecting an item-explanation pair (j, e) to present to the user. We refer to this presentation as the *action*, in line with bandit terminology. Choosing the optimal action $(j^*, e^*) = \arg_{j,e} \max r(j, e, x)$ in any particular context x is the exploitative action, but as explained in Section 1, exploiting at every time step is not optimal in the long run due to the fact that r is only an estimator of the true reward function. We therefore seek a *policy* $\pi(\cdot|x)$ that is a distribution over actions from which the bandit samples each action in response to the context.

With this definition of the problem to be solved we next present Bart, a contextual bandits approach to recommending with explanations.

3.2 Bart

The purpose of Bart is to learn how items and explanations interact within any given context to predict user satisfaction. It does this in a reinforcement learning setting where it must decide which actions to take next to gather feedback. Our overall strategy with Bart is to build an effective model of satisfaction from sparse inputs and to use the contextual bandit framework to manage exploration and training.

The assumptions of the reward model in a standard MAB [Sutton and Barto, 1998] lead to simple procedures for maintaining a reward belief for each action, e.g., by calculating the posterior distribution $p(R|\mathcal{D}_n)$ given a history of action-reward data $\mathcal{D}_n$ or maintaining a mean and variance estimate for each action.

A criticism of the standard MAB is that it ignores contextual information, denoted by X, that could affect the reward. For example, in recommendation, the context may include user features, item features, time of day, platform etc.. This motivates collecting and observing X before each action and making the choice of which action to take dependent on X. This is known as a contextual bandit [Agarwal *et al.*, 2014].

There are three important elements to the contextual bandit: the context, reward model, and exploration-exploitation policy. We describe each below.

Context & Reward Model

The reward model represents the predicted user engagement resulting from an explained recommendation within a given context. As highlighted in Section 3.1, the cardinality of this input space is $J \times E \times \mathcal{X}$ which is far larger than can be explored exhaustively. We therefore must resort to making assumptions about the structure of the reward function r.

One of the simplest assumptions imposes linearity on the reward model,

$$r(j, e, x) = \sigma(\theta_{\text{global}} + \theta_j^\top 1_j + \theta_e^\top 1_e + \theta_x^\top x), \quad (1)$$

where 1_i represents a one-hot vector of zeros with a single 1 at index i. Eq. 1 combined with a cross-entropy loss function is a logistic regression model of a binary reward, such as whether or not an item was consumed.

For convenience, we stack all the inputs and denote them by a single augmented input vector $x' = [1_j^\top, 1_e^\top, x^\top]^\top$ in the same way that we stack all the corresponding parameters $\theta = [\theta_j^\top, \theta_e^\top, \theta_x^\top]^\top$. The logistic regression model is then defined as

$$r^{(1)}(j, e, x) = \sigma(\theta_{\text{global}} + \theta^\top x'), \quad (2)$$

where the aggregated context is now x' and contains information about the item, explanation, and context.

The main disadvantage of logistic regression is that the recommendations are the same for all users due to the linearity assumption. To get more personalized recommendations, we can introduce weighted sums of higher order interactions between elements in the aggregated context vector x'. When the weights are inner products of latent embeddings, the resulting model family is known as the factorization machine [Rendle, 2010],

$$r^{(2)}(j, e, x) = \sigma(\theta_{\text{global}} + \theta^\top x' + \sum_{a=1}^{D} \sum_{b>a}^{D} {v_a}^\top v_b x'_a x'_b) \quad (3)$$

$$r^{(3)}(j, e, x) = \sigma(\theta_{\text{global}} + \theta^\top x' + \sum_{a=1}^{D} \sum_{b>a}^{D} {v_a}^\top v_b x'_a x'_b + \sum_{a=1}^{D} \sum_{b>a}^{D} \sum_{c>b}^{D} < v_a, v_b, v_c > {x'}_a {x'}_b {x'}_c), \quad (4)$$

where D is the dimensionality of x, we introduce latent embeddings v, and $< x, y, z >$ represents the sum of element-wise products of vectors x, y, z. In general, we refer to $r^{(t)}$ as a t^{th}-order factorization machine.

Off-Policy Training

We use counterfactual risk minimization (CRM) to train the contextual bandit [Joachims and Swaminathan, 2016]. CRM uses importance sample reweighting to target the expectation to the uniform policy when training with data from a collection policy π_c, resulting in optimal parameters $(\hat{\theta}, \hat{v})$,

$$\hat{\theta}, \hat{v} = \arg_{\theta,v} \max \mathbb{E}_{A\sim\text{Uniform}(\cdot)}[\mathbb{E}_{X,R}[\log p_{\theta,v}(R|A,X)]] \tag{5}$$

$$\approx \arg_{\theta,v} \max \frac{1}{N} \sum_{n=1}^{N} \frac{\text{Uniform}(a_n)}{\pi_c(a_n)} \log p_{\theta,v}(r_n|a_n, x_n). \tag{6}$$

As an example in a recommender systems context, if the deployed recommender is suggesting and explaining to a large number of users (j_1, e_1) and the same item with a different explanation (j_1, e_2) to a small number of users, then the logged data will reflect these proportions. Training a new model from the logged data will unduly focus on minimizing the error for explanation e_1 at the expense of e_2 even if the latter has substantially more engagement. In a totally linear discrete model, such as $r^{(1)}$ in Eq. 1 when the context consists of one-hot encodings, there is no trade-off. But any non-linear model, such as a factorization machine or neural network, will sacrifice unbiasedness for generalization.

It follows that we must know the collection policy π_c, referred to as the propensity score in causal analysis, in order to do counterfactual training. We consider how to do this next.

Exploration-Exploitation Policy & Propensity Scoring

We finally define the exploration-exploitation approach and collection policy for propensity scoring in Bart.

When presented with context x and user u, the optimal action under the reward function is $(j^*, e^*) = \arg_{j,e} \max r(j, e, x)$. There are several exploration approaches to select from, most based on variants of Thompson sampling, upper confidence bounds, and epsilon-greedy [Sutton and Barto, 1998]. We focus on epsilon-greedy for simplicity of implementation in production and propensities.

A standard epsilon-greedy treatment of the reward function gives equal probability mass to all non-optimal items in the validity set $f(e, x)$ and $(1 - \epsilon)$ additional mass to the optimal action (j^*, e^*).[1] The limitation of this approach is that the policy must either exploit (j^*, e^*) or explore the item and explanation simultaneously. But if the contribution an explanation alone makes to engagement is non-zero (i.e., if $\theta_e \neq 0$ in Eq. 1- 4) then this is overly exploratory as there may exist a set of effective explanations that Bart should focus on regardless of the items.[2] Furthermore, exploring over a large number of actions results in small propensity scores. This has the effect of giving the counterfactual objective in Eq.6 high variance, making training unreliable. Finally, in practice, we are sometimes constrained to presenting multiple items within an explanation at once. This most commonly occurs in the shelf interface on websites.

In light of these considerations, Bart uses conditional exploration. Specifically, it decides whether to explore or exploit the items separately from the explanations, all the while keeping the same underlying reward model r that captures the interactions between items, explanations, and the context,

$$\pi_c^{\text{item}}(j \mid x, e) = \begin{cases} (1-\epsilon) + \frac{\epsilon}{|f(e,u)|}, & \text{if } j = j^*, j \in f(e,x) \\ \frac{\epsilon}{|f(e,u)|}, & \text{if } j \neq j^*, j \in f(e,x) \\ 0, & \text{otherwise.} \end{cases}$$
$$\text{where } j^* = \arg_{j_1} \max r(j_1, e, x) \tag{7}$$

$$\pi_c^{\text{expl.}}(e \mid x, j) = \begin{cases} (1-\epsilon) + \frac{\epsilon}{|\mathcal{E}|}, & \text{if } e = e^*, j \in f(e,x) \\ \frac{\epsilon}{|\mathcal{E}|}, & \text{if } e \neq e^*, j \in f(e,x) \\ 0, & \text{otherwise.} \end{cases}$$
$$\text{where } e^* = \arg_{e_1} \max r(j, e_1, x). \tag{8}$$

Items are sampled first from π_c^{item} for all $e_1 \in \mathcal{E}$. Then, conditional on the items chosen, the explanation is sampled from $\pi_c^{\text{expl.}}$. The overall propensity score is therefore $\pi_c = \pi_c^{\text{item}} \pi_c^{\text{expl.}}$. To further address small propensities giving the objective high variance, combination of normalized importance sample reweighting and propensity score capping is used [Gilotte *et al.*, 2018].

4 Empirical Evaluation

We use logged feedback data and live production traffic from an online music streaming service to evaluate different versions of Bart against benchmarks. In our experiments, we find that personalizing explanations and recommendations provides a significant increase in estimated user engagement, as estimated by offline and online metrics. We start in Section 4.1 with offline evaluations where we test two hypotheses and find that explanations have a significant impact on user engagement. We perform a set of offline experiments that show that considering pairwise and three-way interactions within the context vector is a much more accurate approach for Bart. In Section 4.2 we describe our experiments in an online A/B test, including the challenges involved in building a production scale test. We find that Bart provides more than a 20% improvement in stream rates compared with statically ordering explanations.

4.1 Offline Evaluation

Offline evaluation uses logged user interaction data with randomized recommendations to check hypotheses about explainable recommendations and to compare different recommendation algorithms. A summary of the offline data collected from a streaming mobile music recommendation service is shown in Table 3.

Testing the Explanation Hypothesis

We first test the null hypothesis that explanations have no effect on user satisfaction. Rejecting the null hypothesis means that users respond differently depending on which explanation is used, and if this is the case, it motivates the learning of personalized explanations for users.

[1] If there happens to be two or more equal optimal actions the additional probability mass may be split equally between them.

[2] Decreasing the ϵ parameter does not solve the issue because it only changes the proportion of exploration actions, not what the exploration actions look like.

Table 2: Subset of Shelves Considered in Hypothesis Tests

Explanation	# Impressions
Because it's `[day of week]`	140.3K
Inspired by `[user]`'s recent listening	138.4K
Because it's a new release	140.5K
Because `[user]` likes `[genre]`	130.7K
Because it's popular	140.5K
Mood	140.7K
Focus	140.5K

Null Hypothesis 1 *The outcome of whether an explanation results in user stream is independent of choice of explanation.*

We use a χ^2 test to determine whether the null can be rejected by the offline data at 1% significance. We focus on the top 7 explanations to ensure that there were no population differences between the groups that see each explanation. In doing so, we account for the candidate selection step outlined in Section 4.1. The 7 explanations examined here are listed in Table 2. There are 6 degrees of freedom to the test, giving a threshold of $\chi^2_{1\%,6} = 22.458$. The data gives $\chi^2 = 570.67$. Therefore, we reject the null hypothesis and find that stream behavior is dependent on the choice of explanation for the 7 popular explanations in our data set.

Could the difference in stream probability be due to the fact that different explanations have different valid items? In other words, the fact that explanation A has higher engagement than explanation B could be because users respond better to A or because A has a more engaging set of items to recommend (or both). We investigate whether there is an intrinsic appeal to some shelves for some users with the next null hypothesis.

Null Hypothesis 2 *The outcome of whether a user streams from a recommendation is independent of the choice of explanation provided for that recommendation.*

To check Null Hypothesis 2, we focus on the subset of playlists that appear in more than one explanation. There were two sets of explanations that exhibited significant playlist overlap. The first set consisted of $set_1 = \{$ *day-of-the-week, inspired-by-recent-listening, because-user-likes-genre, popular* $\}$ resulting in a $\chi^2_{3,1\%}$ threshold of 16.266 at 3 degrees of freedom. The second set consisted of $set_2 = \{$ *mood, focus* $\}$ resulting in a $\chi^2_{1,1\%}$ threshold of 10.828. We set the significance threshold for both tests at 1%.

The data indicate $\chi^2 = 126.85$ for set_1 and $\chi^2 = 0.11236$ for set_2, meaning that we can reject the null hypothesis for set_1 but not set_2. Therefore, we know that explaining the same playlist as recommended because it is popular vs. because the user likes a certain genre vs. because it's linked to the day of week vs. because it is similar to what the user has been listening to recently makes a difference to engagement. Specifically, popularity was the weakest driver of engagement while related to recent listening and genre-based explanations were the strongest. [3]

[3] An alternative reason why recent listening might have stronger engagement, even with the same playlists, is because the recommendations are presented at the right *time*. While this likely contributes to the effect, the evidence for rejecting Null Hypothesis 2 is strong enough that this additional effect does not change our conclusion.

Table 3: Summary of Test Data

Experiment Group	# Impressions	# Users	# Items
Hypothesis testing	970K	140K	140K
Offline experiments	190K	8.6K	9.6K
Online experiments	3.8M	560K	230K

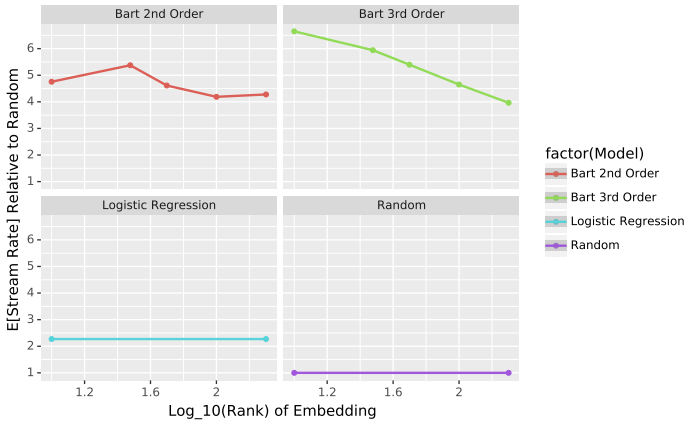


Figure 2: Experimental results on offline music recommendation data show that a model that captures three-way interactions between contextual features outperforms other methods. Logistic and Random have no embeddings and are therefore constant in that dimension.

Comparative Offline Experiments

We compare the impact that various recommendation methods have on user satisfaction estimated from logged randomized interaction data. We split the logged data into three sets: the training set comprising the first week of the data, the validation set comprising one day, and the test set comprising the next week. A summary of test data is given in Table 3. We used the whole training week but randomly subsampled 1% of the test week so the training data sizes are approximately 100x those of the test set. The recommendation methods, metrics, and results are discussed next.

Recommendation Methods We compare different versions of Bart against baselines. The methods we consider in the offline experiments are,

- **Bart 2nd Order** uses Bart with a 2nd order factorization machine reward function $r^{(2)}$ given in Eq. 3.
- **Bart 3rd Order** uses Bart with a 3rd order factorization machine reward function $r^{(3)}$ given in Eq. 4.
- **Logistic Regression** provides a static ordering of the items and explanations according to their individual controlled popularity defined in Eq. 2.
- **Random** randomly selects items and explanations while respecting the validity function. The quality of random

shuffle is lower bounded because the set of recommendations was limited to relevant items.

Metrics We use the expected stream rate, which is an offline estimate of the average number of times a user streamed at least one song from a recommended playlist per recommendation. We use inverse propensity score (IPS) reweighting to estimate the expectation [Agarwal *et al.*, 2014]. NDCG@K was also considered and we found that it gave very similar results so we omit them here.

Results Figure 2 shows the expected stream rate for the recommendation methods under a range of embedding sizes and order of interactions. We find that Bart with a reward model that considers third order interactions between elements of the context vector performs the best across a variety of embedding sizes and is typically 5 times better than random as measured by stream rate. The reward model that considers second order interactions performs almost as well, but is more sensitive to the size of the embedding and appears to peak at embedding size 30. We see no benefit to increasing the embedding size beyond 30 for any approach considered because larger embeddings are vulnerable to overfit the training data.

4.2 Online A/B Tests

We now evaluate Bart in an online A/B test with live production traffic from an online music recommender service. For this test we introduce the Control benchmark that ranks the explanations statically according to their engagement but uses a default ordering of items. In this section we start by discussing the production challenges related to this test, describe the control experiment and contextual features considered, then discuss the results.

Production Challenges There are several challenges we faced moving Bart into a production setting for online A/B testing. Firstly, too much exploration has the potential to confuse users. To address this, we limited the exploration rate in our experiments to 10%, giving significant probability mass to all actions in the space while reducing the negative effects of exploration. Secondly, prediction has strict latency requirements. This constrained the set of features we can use. Thirdly, the features had to be available in both batch (offline) and real time settings. Some features are hard to make available in real time, such as the set of all artists a user has ever listened to. Other features are hard to make available in batch, such as the last 15 songs listened to. Fourthly, it is imperative that the feature extraction and processing is identical during both training and production. It is easy for inconsistencies to be introduced if these occur in different languages or systems.

Results The results in Figure 3 mostly agree with the offline experiments from Figure 2. Considering two way and three way interactions with Bart provides a considerable boost over the Logistic Regression, Control, and Random methods. Since the confidence intervals for Bart 2nd Order and Bart 3rd Order overlap, we cannot say that one has a higher stream rate than the other here.

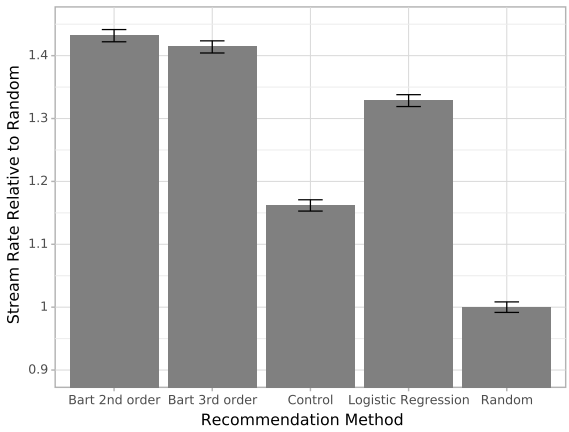


Figure 3: Online results from a music recommender system. Error bars indicate the 95% confidence interval.

5 Conclusions & Future Work

In this paper we introduced Bart, an algorithm for jointly personalizing recommendations and associated explanations for providing more transparent and understandable suggestions to users. Bart is an efficient method for addressing the exploration-exploitation problem in recommendation. Experiments show that explanations affect the way that users respond to recommendations and that Bart significantly outperforms the best static ordering of explanations.

References

[Agarwal *et al.*, 2014] Alekh Agarwal, Daniel Hsu, Satyen Kale, John Langford, Lihong Li, and Robert Schapire. Taming the monster: A fast and simple algorithm for contextual bandits. In *International Conference on Machine Learning*, pages 1638–1646, 2014.

[Chaney *et al.*, 2017] Allison JB Chaney, Brandon M Stewart, and Barbara E Engelhardt. How algorithmic confounding in recommendation systems increases homogeneity and decreases utility. *arXiv preprint arXiv:1710.11214*, 2017.

[Friedrich and Zanker, 2011] Gerhard Friedrich and Markus Zanker. A taxonomy for generating explanations in recommender systems. *AI Magazine*, 32(3):90–98, 2011.

[Gilotte *et al.*, 2018] Alexandre Gilotte, Clément Calauzènes, Thomas Nedelec, Alexandre Abraham, and Simon Dollé. Offline a/b testing for recommender systems. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 198–206. ACM, 2018.

[Joachims and Swaminathan, 2016] T. Joachims and A. Swaminathan. Tutorial on counterfactual evaluation and learning for search, recommendation and ad placement. In *ACM Conference on Research and Development in Information Retrieval (SIGIR)*, pages 1199–1201, 2016.

[Joachims *et al.*, 1997] Thorsten Joachims, Dayne Freitag, Tom Mitchell, et al. Webwatcher: A tour guide for the

world wide web. In *IJCAI (1)*, pages 770–777. Citeseer, 1997.

[Kangasrääsiö *et al.*, 2015] Antti Kangasrääsiö, Dorota Glowacka, and Samuel Kaski. Improving controllability and predictability of interactive recommendation interfaces for exploratory search. In *Proceedings of the 20th international conference on intelligent user interfaces*, pages 247–251. ACM, 2015.

[Kouki *et al.*, 2017] Pigi Kouki, James Schaffer, Jay Pujara, John O'Donovan, and Lise Getoor. User preferences for hybrid explanations. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*, pages 84–88. ACM, 2017.

[Lamere, 2017] Paul Lamere. https://twitter.com/plamere/status/822021478170423296. Twitter, 2017.

[Li *et al.*, 2010] Lihong Li, Wei Chu, John Langford, and Robert E Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, pages 661–670. ACM, 2010.

[Rendle, 2010] Steffen Rendle. Factorization machines. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pages 995–1000. IEEE, 2010.

[Shani *et al.*, 2005] Guy Shani, David Heckerman, and Ronen I Brafman. An mdp-based recommender system. *Journal of Machine Learning Research*, 6(Sep):1265–1295, 2005.

[Srivihok and Sukonmanee, 2005] Anongnart Srivihok and Pisit Sukonmanee. E-commerce intelligent agent: personalization travel support agent using q learning. In *Proceedings of the 7th international conference on Electronic commerce*, pages 287–292. ACM, 2005.

[Sutton and Barto, 1998] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[Swaminathan *et al.*, 2017] Adith Swaminathan, Akshay Krishnamurthy, Alekh Agarwal, Miro Dudik, John Langford, Damien Jose, and Imed Zitouni. Off-policy evaluation for slate recommendation. In *Advances in Neural Information Processing Systems*, pages 3635–3645, 2017.

[Tintarev and Masthoff, 2007] Nava Tintarev and Judith Masthoff. A survey of explanations in recommender systems. In *Data Engineering Workshop, 2007 IEEE 23rd International Conference on*, pages 801–810. IEEE, 2007.

Declarative Description: The Meeting Point of Artificial Intelligence, Deep Neural Networks, and Human Intelligence

Z. Á. Milacski, K. B. Faragó, Á. Fóthi, V. Varga, A. Lőrincz
Faculty of Informatics, Eötvös Loránd University, Budapest, Hungary

Abstract

Advances in deep learning have opened up new ways in integrating algorithms for highly demanding artificial intelligence tasks. In this position paper we propose an architecture that uses the nomenclature of neuroscience to combine AI methods and aims to reach a reasonable explanatory power. According to neuroscience, declarative memory is made of two components; the semantic component that has minor temporal relations if any and the episodic component, which is extended in time and is local both in space and time. We model semantic memory with '*Bag-Models*'. In our model, episodic memory relies on the concept of sparseness, since start and end intervals of any episodes are relatively short and are thus sparse. In turn, we exploit *Sparse Methods* in searches for the time limits of episodes. A '*Consistency Seeking Unit*' (CSU) uses available outputs from both memory types and produces the most likely output. In turn, CSU can give an explanation about the output in terms of the causes, the semantic components, and in terms of their spatio-temporal interaction, the most likely episodes that occurred, are ongoing, or are to occur. We illustrate the framework on the MPII cooking database.

1 Introduction

Artificial intelligence (AI) solutions are present in our daily lives and they keep improving at an unprecedented pace. This is due to the fast development of deep neural networks, being capable for superhuman performance as popularized at TED Talks and in blogs by leading companies, such as Facebook, Google, and Microsoft.

Crowdsourcing based collection of input-output pairs paved the way of recent advances in Deep Neural Networks (DNNs) giving rise to superhuman performances in regression and classification. Although achievements concern narrow and separated fields, still, they led to an explosion in AI. For reviews on this matter, see, e.g. [Schmidhuber, 2015] and the cited references therein.

Nonetheless, considering the possible scope of Big Data, we should note the limitations of human capabilities and crowdsourcing efforts. One may want to look for more efficient methods and in this regard, the need to understand what DNNs did, has been recognized by DARPA [Gunning, 2016]. Recently, AI and Augmented Cognition are also in the focus of interest[1].

As a first step, consider zero-shot or one-shot learning, i.e., solving a task without any [Romera-Paredes and Torr, 2015; Xian *et al.*, 2017] or with only a few [Li *et al.*, 2006] supervisory training examples of that task, respectively. This is Biederman's [1987] '*recognition by components*' theory topped with a Bayesian approach. Conservation laws of physics can also support one-shot learning, e.g., via optical-flow based tracking. This method has its origins in the cca. 100 years old '*Gestalt Principles*' [Köhler, 1929] but includes only a subset of the many rules. Furthermore, human intelligence incorporates information about physical laws, e.g., by exploiting three-dimensional vision. In case of images or videos, depth information is of high value for proper interpretation. Novel methods for the estimation of depth are being developed for both of them, see, e.g., [Vijayanarasimhan *et al.*, 2017].

Beyond such spatial information pieces, temporal methods are also needed for prediction. Temporal processes may be segmented, enabling the estimation of spatially and temporally restricted chunks on multiple scales. These chunks may follow each other, may collide or merge, and may be concurrent, too. Only a few out of them may occur at a time and according to Heraclitus[2] any episode may occur only once. Leaving aside some of the details and considering similar episodes identical, episodes may occur more than once, but still they remain sparse. Furthermore, the starting and ending time intervals of any episodes are relatively short and thus they are even sparser. In addition, the start of an episode is itself an episode, and episodes are interlinked, consider for example the episode of the signal of the starting pistol in 100 metres sprint, then the start of the motion, the steps, the changes in the order of the runners forming a longer episode of about 100 seconds. In this temporal dimension we shall exploit sparse methods.

We aim at explainable AI and consider that the target of explanation is us. It then follows that explanations should match our constraints that originate from our memory sys-

[1] `https://hub.ki/groups/gmuai/overview`

[2] "You cannot step into the same river twice"

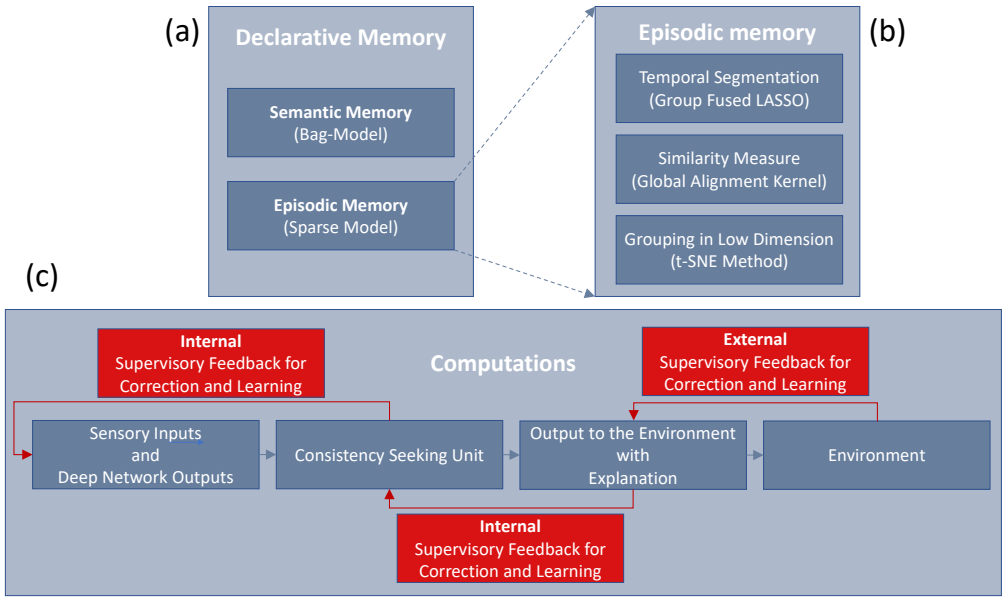


Figure 1: **Model architecture for declarative description.** (a): 'Declarative Memory' is made of two components; 'Semantic Memory' and 'Episodic Memory'. (b): 'Episodic Memory' can be learned by three algorithmic components: 'Temporal Segmentation', 'Similarity Measurement', 'Grouping in Low Dimension'. (c): Computations in the model. Outputs of sensors and deep networks are analyzed by the 'Consistency Seeking Unit' (CSU). CSU provides information to the environment that may include the explanation. Feedforward 'communications' are denoted by grey arrows. Feedback 'information' flow is denoted by red arrows. Feedback can be either internal or external and can serve the upgrading of the representation as well as the tuning of the deep networks.

tem. Figure 1 depicts the architecture that follows the information provided by neuroscience and psychology. According to the knowledge of those fields, there are two types of memories, namely declarative and procedural ones that can be dissociated and only the former can be told and explained (for a recent view see, e.g., [Buzsáki and Moser, 2013] and the cited references therein). In turn, we should be concerned with declarative memory.

Declarative memory is also made of two components, see also the cited work [Buzsáki and Moser, 2013]; semantic and episodic memory. Semantic memory assumes a collection of facts and concepts, independently of the spatio-temporal context. Episodic memory on the other hand represents experiences and events in a mix of concurrent and serial items. Components and their working are shown in Fig. 1

Algorithms for semantic memory have been developed in the literature and we shall refer to those. Episodic memory, however, is a less studied area and we put forth a model that can discover episodes and sub-episodes and can visualize the part of the episodic structure for us: we want to have an explanation or reasoning.

The paper is organized as follows: In this paper we put the emphasis on the formation of episodic memory. For the sake of completeness and comparison, first, we glance through the concept of semantic memory (Sect. 2). Section 3 presents briefly the deep neural networks that we combine, the self-organizing and unsupervised methods as well as the data set we use. We chose mostly from the field of natural language processing, but add a few other examples, too. Illustrations on episodic memories are presented in Sect. 4. It is followed by the discussion (Sect. 5) that includes further possibilities and limitations. Here, we return to the double sided memory system and consider concurrent episodes as semantic memory. Conclusions are drawn in Sect. 6.

2 Brief review of Semantic Memory methods

Semantic memory is about components that belong or involve each other involving facts, concepts, and rules among others. It concerns general knowledge instead of particular autobiographic experiences. In the area of natural language processing (NLP), such general knowledge is typically represented by semantic networks. From the algorithmic point of view, co-occurrences define the network. Such co-occurrences of words in documents, sometimes including the relative frequencies of the words, are called the 'Bag-of-Word' representation. They have been generalized to 'Bag-of-n-Grams' (see, e.g., [Wang *et al.*, 2007] and the cited references therein) and different versions are used in diverse applications.

In the field of NLP, distributional semantics based on the distributional hypothesis tries to capture the idea: words that occur in similar contexts tend to have similar meanings.

In psychology, the concept of 'recognition by components' is similar, since it neglects both spatial and temporal information [Biederman, 1987].

The concept of co-occurrences has been transferred to general visual categorization problems and is called the 'Bag-of-Keypoints' method [Csurka *et al.*, 2004]. We note that components are also hierarchical and meaning might be modified by a larger context as shown in Fig. 2

Figure 2: **Context may modify meaning.** Left hand side: deflated soccer. Right hand sides: table and chairs

Semantic memory is easy to train for generative sparse representation (see later), it is a one shot learning process: training samples form the columns of the generative matrix. Furthermore, high quality guesses become possible for novelties given their context [Pintér *et al.*, 2015]. Importantly, the slowness of the processing can be counteracted by deep learning methods [Sprechmann *et al.*, 2015; Lőrincz *et al.*, 2016].

3 Methods

3.1 Supervised Learning

The architecture contains state-of-the-art DNNs that are pretrained by means of a database of input and desired output pairs. The different outputs may support or may contradict to each other. The tools we used are briefly reviewed below.

Convolutional Pose Machine (CPM)
For estimating human body keypoints on video frames, we exploited the OpenPose[3] [Simon *et al.*, 2017] version of the CPM. It is a fully convolutional network that was trained to localize anatomical landmarks of the human body on single images, including wrist and elbow coordinates, together with their confidence levels.

Optical Flow (OF)
OF is a type of motion detector, which approximates the pixel shifts between two images of identical sizes (for a review, see [Fortun *et al.*, 2015] and the cited references). We applied the recently developed deep neural network variant called FlowNet 2.0[4] [Ilg *et al.*, 2017] in our experiments.

Consistency Seeking Block
In our model, cognition is restricted to *consistency seeking* that can exploit rules and (fuzzy) logic, probabilities and Bayesian inferences, or other methods in order to improve the representation. As a point of reference, a recent paper of Lőrincz et al. [2018] provides illustrative examples. Consistency seeking may incorporate human feedback and machine learning methods as well.

Generative System
The concept may be traced back to 1995 [Hinton *et al.*, 1995] and it underwent considerable evolution in recent years [Goodfellow *et al.*, 2014]. It can derive an estimated state, from which error can be computed, facilitating parameter tuning.

[3] https://goo.gl/gMkoU3
[4] https://goo.gl/Sbtjjb

3.2 Unsupervised Learning

Structure searches promote classification and clustering. We are looking for multi-scale episodes in the presence of outliers. Outliers can't be typical and thus they are sparse.

Robust Principal Component Analysis (RPCA)
RPCA is a well known tool [Candès *et al.*, 2011], an extension of Principal Component Analysis (PCA). We use it for separating slowly changing spatio-temporal signals from the rest. RPCA assumes that the residuals are sparse, offering the separation of episodes on different time scales: RPCA can be used as a step-wise method. We used a Python implementation of RPCA[5] based on the inexact Augmented Lagrange Multiplier solver [Lin *et al.*, 2010] with $\lambda = 0.005$.

Temporal Segmentation using Group Fused LASSO (GFL)
In the case of episodic descriptions and learned models that approximate the individual episodes, residuals, or model errors can be derived. Part of the error is Gaussian noise and has no further information. Another part can signal anomalies, such as switches, e.g., that the current model is finished or that a new sub-episode started. The GFL method [Tibshirani *et al.*, 2005; Bleakley and Vert, 2011] assumes that such sparse changes can slice signals across time. We applied it for detecting change-points co-occurring across the dimensions of multivariate signals. The method approximates the original signal and imposes a regularization constraint on the multivariate total variation, leading to piece-wise polynomial. In the case when a model is available, the average error should be about zero and thus a constant approximation is suffice. The trade-off constant $\lambda > 0$ controls the relative strength of the costs and thus the frequency of the *sparse* switches. In the experiments carried out in this work, we applied the CVXPY [Diamond and Boyd, 2016] convex toolbox with SCS solver [O'Donoghue *et al.*, 2016]. We let $\lambda \in \{8, 64\}$.

Segment Similarity using Global Alignment (GA) Kernel
Dynamical properties of different systems may be similar but can still happen at different paces and with diverse delays. Comparison of episodes with switching dynamics thus calls for robust time warping. We calculated time series similarity for pairs of windows of segmented sequences – the so-called Gram matrix – using the Global Alignment (GA) kernel [Cuturi, 2011]. This procedure, like Dynamic Time Warping, matches time steps between the two series, but instead of taking the best alignment it weighs all alignments avoiding greedy selection by computing the soft-minimum. We used the Python implementation of the GA kernel[6]. We used the scale parameter $\sigma = 1$.

Low-Dimensional Embedding (LDE) using t-SNE
t-Distributed Stochastic Neighbor Embedding (t-SNE) [van der Maaten and Hinton, 2008] is an LDE scheme for high-dimensional data, suitable for discovering new knowledge (in the form of clusters) in an unsupervised fashion. It can preserve local structure and uncover the

[5] https://goo.gl/msNDst
[6] https://goo.gl/Z5wCM6

global one as multi-scale clusters. It constructs a probability distribution over sample similarities (GA kernel values in our experiment) both in the high- and low-dimensional space and minimizes the divergence between the joint probabilities. We used the scikit-learn implementation[7] with perplexity value 50. Again, the underlying assumption is a kind-of-sparsity, now in the form of the heavy-tailed Student's t-distribution.

3.3 Data set

The MPII Cooking Activities Data set[8] consists of videos depicting subjects preparing food in kitchen, while interacting with various objects. Similar scenarios are played by different people in a realistic setting and the videos are annotated by experts based on the ongoing low-level activities and the objects used [Regneri *et al.*, 2013].

For our experiments – temporal segmentation of cooking activities and clustering of the episodes – we extracted wrist and wrist-minus-elbow joint coordinates for both arms (i.e., 8 features) with CPM. For GA computation and t-SNE embedding of fixed 100 frames long segments with 50 frames overlap we used 24 full-length videos, the ones with dish id 02, 12, 21 and 23.

4 Illustrative Examples

Our thoughts concern sparsification methods that we utilize for episodic descriptions. We start by presenting some results on the MPII database. We also illustrate consistency seeking, although in a fairly limited manner, and provide examples on clustering, too.

4.1 RPCA

RPCA is an efficient sparse outlier seeking procedure that can work online. The procedure may deal with different temporal durations. RPCA can disentangle the background (the slow part of the input) from the foreground (the outlier). In turn, the kitchen of the MPII database is separated from the person entering it (Fig. 3). Note that background is almost left intact as if the foreground – i.e., the residual of the RPCA procedure – were transparent.

Upon tracking the person and analyzing the residual of the previous stage, the motion becomes visible in the residual of RPCA at this second stage (lower sub-figure of Fig. 3). This way one can find spatially restricted episodes and the belonging sub-episodes they have.

4.2 Group Fused LASSO

GFL is also a sparse component seeking procedure. We demonstrate its working in a simple case. The algorithm searches for approximately constant temporal signals covered by outliers and sparse switches between such constant-like dependencies. Figure 4(a) depicts the temporal segmentation obtained by this algorithm for different degrees of sparsification (different λ values) on full-length video 's08-d02-cam-002'. The original signal containing CPM features is shown in the top row. Depending on the value of λ, the procedure finds different *sub-episodes* shown by the two middle rows. Changes of the activity labels provided by experts (bottom row) correlate with those in the segmentation(s). Note that segments have different lengths. We shall also treat an example using identical length segments later.

[7] https://goo.gl/n5zS31

[8] https://goo.gl/EZLrAJ

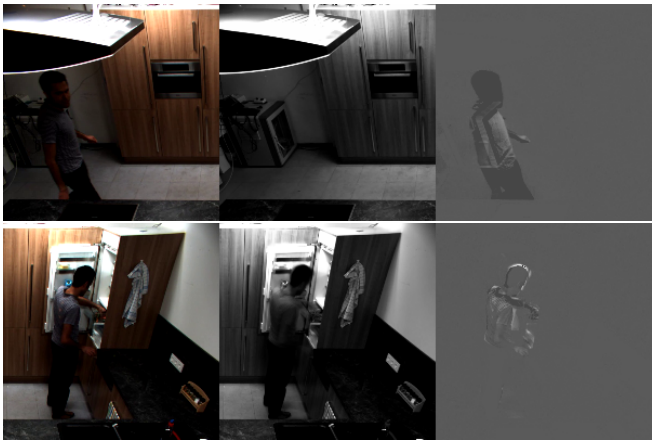


Figure 3: **'Episodes' with RPCA.** Columns from left to right: original image, low-rank component (kitchen background in top row and person in bottom row), sparse component (person walking in top row and smaller motions while standing in bottom row). The top row shows the higher-level episode, i.e., the kitchen scenario itself, while the bottom row reveals sub-episodes, i.e., the person taking out items from the fridge. The brightness of the residuals has been increased for better visibility.

4.3 t-SNE

We computed similarities between the 100 frames long segments using the GA kernel, as shown in Fig. 4(b). The GA induced distances were then visualized with t-SNE in 2D (Fig. 4(c)). In the embedding, each color point represents a single 100 frames long segment. Points belonging to 'peel' (yellowish green) are clustered together to a considerable extent. On the other hand, 'wash' (pink), 'cutoffends' (pale green) and 'throwinggarbage' (green) were also near.

For the sake of illustration, we proceed by connecting temporally adjacent points by lines in order to reveal temporal order on video '*s10-d02-cam-002*' in the inset of Fig. 4(c). The inset illustrates that spatial and temporal closeness can be used to cluster sub-episodes of similar nature, for the respective motions of 'peel', 'wash' and 'cutoffends' in the present case, while, avoiding activities like 'throwinggarbage'.

We note that further distinction may be possible by exploiting additional information about the nearby objects (e.g., tap) or objects held in the hand (e.g., peeler). Such additional information can be gained from detecting 'objectness' [Alexe *et al.*, 2012] and the proximity of the positions (sampled points of the trajectory) it follows and from human partners upon asking about the name of the object(s).

4.4 Consistency Seeking

In the supervised learning approach, the block named 'consistency seeking' (Fig. 1) represents the cognition stage. This stage may include human knowledge, such as laws of physics

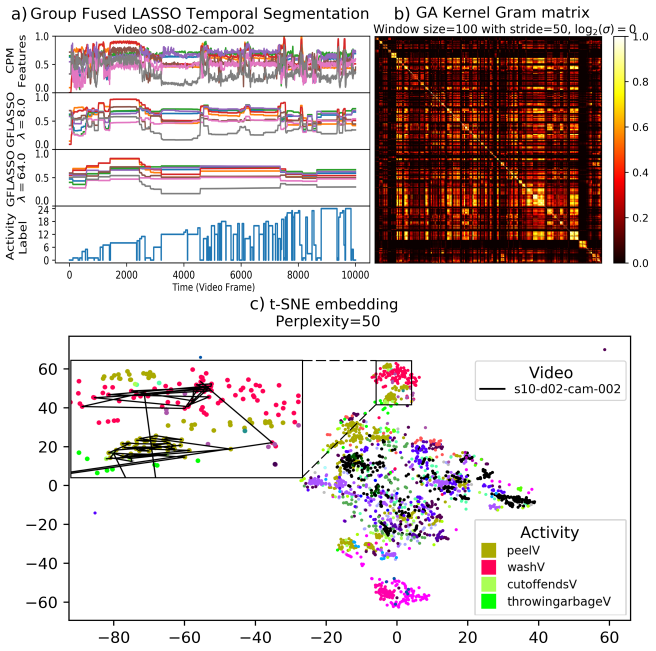


Figure 4: **Unsupervised learning results.** (a) Temporal Segmentation using Group Fused LASSO. Top and bottom rows: original signal and ground truth labels, respectively. Middle rows: segmentation with $\lambda \in \{8, 64\}$. The obtained segments correlate with the ground truth Activity labels. (b) GA Kernel Gram matrix shows the similarities of 100 frames long segments with kernel parameter $\sigma = 1$. (c) t-SNE embedding of GA Kernel similarities with perplexity 50. The embedding yields clusters that correlate with the ground truth Activity labels. The inset shows a trajectory of consecutive segments within a video. Trajectories are composed of changes in directions (shorter sub-episodes) that cluster and determine longer sub-episodes. Closeness in space and temporal connectedness can be used for defining clusters that can be assigned symbol labels. The inset in (c) shows that human annotation shown by colors and clusters indicated by spatio-temporal relatedness tend to match each other. In turn, symbol labels need to be agreed by the machine and the user. Best viewed in color.

and correlations found by the machine, for example. Consistent representation is then a non-contradicting and most likely set of outputs. This output may overwrite incorrect partial results, including the inputs of deep networks.

We illustrate the interplay by hand pose estimation. It may fail, as shown in Fig. 5. However, Optical Flow (OF) can uncover the mismatches in time and could be used to override the output. Furthermore, the new corrected samples may serve fine tuning of the CPM parameters.

Activities in the cooking scenario are composed of smaller concurrent and possibly overlapping sub-episodes. In particular, vegetable peeling, washing, or knife sharpening involve similar periodical movements. Peeling can be distinguished from the other two examples, since small pieces are falling down and can be detected by OF, see Fig. 6. The knowledge that something 'falls down' may require human knowledge unless physics projected to 2D have been learned. In any case, the small piece moving separately is a new concurrent episode, which does not happen during knife sharpening and

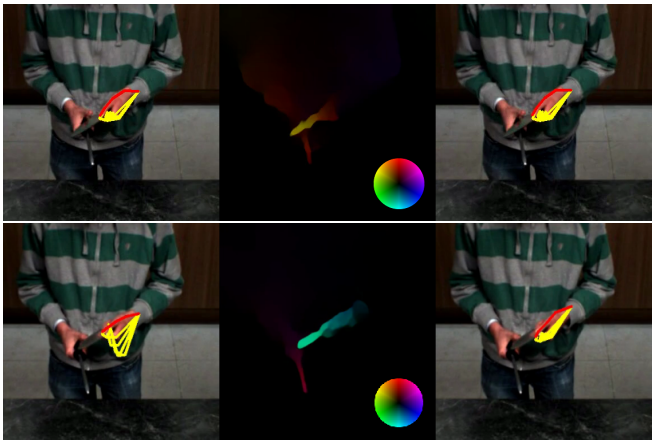


Figure 5: **Correcting the outputs of the Convolutional Pose Machine (CPM).** Top row: knife sharpening at a given time (lhs), at one frame later (rhs) and the Optical Flow (OF) between. Bottom row: the same one second later (lhs and middle) and 1 second with OF estimation alone (rhs). Hand pose detection is erroneous (lhs of the bottom row) for most frames during this second (not shown). OF based tracking, however, remains precise and may be used for self-improvement. Columns Colored circle in each row refers to the color coding of the direction and speed.

can be noted.

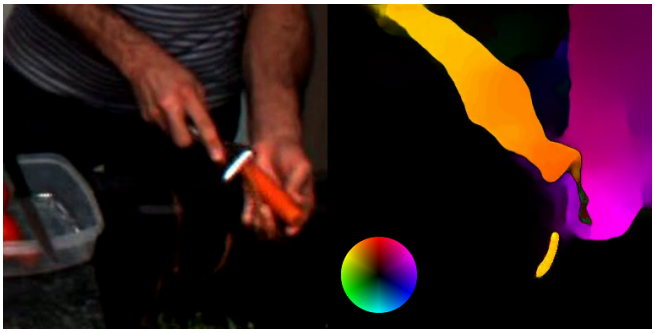


Figure 6: **Sub-episode revealed by Optical Flow (OF).** Carrot peeling involves periodical movements, but OF also reveals the falling peels barely visible in the video. This can be considered an anomaly and can be recognized as a novel sub-episode. Left: original frame. Right: OF from previous frame. Colored circle represents the color coding of the direction and speed.

5 Discussion

XAI is a challenge having a long history: we want to know why and how a neural network came to a decision? We put forth concept of declarative memory defined by neuroscience and psychology, since (i) it seems the only way for providing explanations and (ii) the advance of deep learning and sparse methods enabled us to provide promising set of algorithms that machine learning may apply. We illustrated the working of the algorithms on the MPII cooking scenario.

Declarative memory is made of two components. The first one is semantic memory, having little or no temporal dependency. We mentioned that (structured) sparse methods seem

useful, since they can be trained by one-shot learning and then fast processing can be achieved internally by self-supervised training of deep networks [Sprechmann *et al.*, 2015; Lőrincz *et al.*, 2016]. We also noted that 'Bag-Models' are similar in nature and they have been been used in diverse fields, including NLP and visual categorization.

As a further note, ambiguities can be resolved by knowing/recognizing (i) the details, i.e., the components [Biederman, 1987] and (ii) the context as well. In turn, semantic memory requires more, possibly many deep networks.

Since not all information pieces may be available at a time, different interpretations may co-exist calling for probability based considerations and maximum likelihood estimations. This is the way that XAI becomes efficient: it can explain the details of a classification given the probability estimates of the components and/or the sub-episodes.

The critical part of our approach is the episodic segmentation and clustering of the episodes. The former is made possible by the recent advances in ℓ_1-Magic, see, e.g., [Candès and Romberg, 2005] and the references cited. For the latter, we used LDE into 2D to enable communication with human observers by means of arguments about closeness in space and time and large jumps between such domains.

Detection of anomalies suggest searches for cause-effect relations, whereas regularities in episodic sequences give rise to deeper understanding and longer term predictions. We showed a few examples. Vegetable peeling in the cooking scenario is an episode that involves smaller periodical movements as sub-episodes. However, using OF, falling pieces can be observed, as illustrated by Fig. 6. This is a novel sub-episode, giving rise to a multi-scale segmentation. Note that this new knowledge might be included in the feature set and used for temporal segmentation via iterated application of the unsupervised part of our architecture. One may end up with better episode clusters after the LDE stage, where the washing and peeling activities are not confused anymore, as seen on Fig. 4(c).

We note that in goal oriented systems explanation should concern past experiences, ongoing processes and intentions, too. This type of explanation includes future plans, expected changes in the environment and thus it should be model based. Episodic description seem to fit this direction [Tősér and Lőrincz, 2015], but it is beyond the scope of the present work.

We note that generative networks can be used for error computation and can serve the tuning of the deep networks via error backpropagation. This is the direction of end-to-end learning. Our architecture raises end-to-end learning to a higher level via spatio-temporal component segmentation followed by component based consistency seeking and, finally, by consistency based self-improvements. In addition, the homunculus fallacy can be eliminated by means of generative networks [Lőrincz *et al.*, 2002].

Last, but not least, concurrent episodes are like components of a larger episode and thus they may also be part of semantic memory. Consider the sub-components of 100 metres sprint, mentioned in the introduction, and that it is part of the Olympic Games, no matter if athletics comes first or is placed to the end of the Games.

6 Conclusion

In this position paper, we introduced an architecture that combines deep neural networks and unsupervised machine learning techniques. The architecture is based on concepts of neuroscience and psychology in order to promote machine explanations *in our way*. We started from declarative memory and its two components, semantic memory that we reviewed quickly and episodic memory, the main subject of our illustrations.

Our approach enables learning by means of (i) temporal segmentation, (ii) finding regularities in temporal series, e.g., with OF or by other known methods, such as prediction by partial matching, for example. Higher and lower order semantic descriptions can lead to multi-scale episodic decomposition. They may also improve both predictive and discriminative capabilities.

Deep learning is an important part of our approach. We note that upon tuning, they become a kind of trainable and adaptive sensors that may be erroneous. This thought emphasizes the need for consistency seeking units that may also serve self-training. Self-training have also appeared in semantic memory capable of one-shot learning followed by self-supervision of deep learning tools for fast processing [Lőrincz *et al.*, 2016].

We consider the segmentation of episodes followed by their clustering in low dimensions as an important tool for visual inspection based explanation in XAI systems.

We close by noting that a proper benchmark for proving the efficiency of our sparsity related concepts are not available yet. We restricted ourselves to illustrations. The approach has its merits in the mathematical backing on the field of sparsity and in the motivation from neuroscience and psychology.

Acknowledgments

The research has been supported by the European Union, co-financed by the European Social Fund (EFOP-3.6.2-16-2017-00013, EFOP-3.6.3-16-2017-00001). The authors would like to thank Róbert Adrian Rill for his careful reading of the manuscript and his constructive suggestions for improving it.

References

[Alexe *et al.*, 2012] B. Alexe, T. Deselaers, and V. Ferrari. Measuring the objectness of image windows. *IEEE Tr. PAMI*, 34(11):2189–2202, 2012.

[Biederman, 1987] I. Biederman. Recognition-by-components. *Psych. Rev.*, 94(2):115, 1987.

[Bleakley and Vert, 2011] K. Bleakley and J. P. Vert. The group fused lasso for multiple change-point detection. *arXiv:1106.4199*, 2011.

[Buzsáki and Moser, 2013] Gy. Buzsáki and E. I. Moser. Memory, navigation and theta rhythm in the hippocampal-entorhinal system. *Nature Neurosci.*, 16(2):130, 2013.

[Candès and Romberg, 2005] E. Candès and J. Romberg. l_1-magic: Recovery of sparse signals via convex programming. `https://goo.gl/3eVQCx`, 2005.

[Candès *et al.*, 2011] E. J. Candès, X. Li, Y. Ma, and J. Wright. Robust principal component analysis? *Journal of the ACM (JACM)*, 58(3):11, 2011.

[Csurka *et al.*, 2004] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. In *Workshop Stat. Learn. Comp. Vis., ECCV*, volume 1, pages 1–16, 2004.

[Cuturi, 2011] M. Cuturi. Fast Global Alignment Kernels. In *ICML*, pages 929–936, 2011.

[Diamond and Boyd, 2016] S. Diamond and S. Boyd. CVXPY: A Python-embedded modeling language for convex optimization. *JMLR*, 17(1), 2016.

[Fortun *et al.*, 2015] D. Fortun, P. Bouthemy, and C. Kervrann. Optical flow modeling and computation: A survey. *CVIU*, 134:1 – 21, 2015.

[Goodfellow *et al.*, 2014] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *NIPS*, pages 2672–2680, 2014.

[Gunning, 2016] D. Gunning. Explainable artificial intelligence (XAI), 2016. Tech. Rep. DARPA-BAA-16-53.

[Hinton *et al.*, 1995] G. E. Hinton, P. Dayan, B. J. Frey, and R. M. Neal. The "wake-sleep" algorithm for unsupervised neural networks. *Science*, 268(5214):1158–1161, 1995.

[Ilg *et al.*, 2017] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox. Flownet 2.0. In *CVPR*, pages 1647–1655, 2017.

[Köhler, 1929] W. Köhler. Gestalt Psychology. *New York: Horace Liveright*, 1929.

[Li *et al.*, 2006] FF. Li, R. Fergus, and P. Perona. One-shot learning of object categories. *IEEE Tr. PAMI*, 28(4):594–611, 2006.

[Lin *et al.*, 2010] Z. Lin, M. Chen, and Y. Ma. The Augmented Lagrange Multiplier Method for Exact Recovery of Corrupted Low-Rank Matrices. *arXiv:1009.5055*, 2010.

[Lőrincz *et al.*, 2002] A. Lőrincz, B. Szatmáry, and G. Szirtes. The mystery of structure and function of sensory processing areas of the neocortex: a resolution. *J. Comput. Neurosci.*, 13(3):187–205, 2002.

[Lőrincz *et al.*, 2016] A. Lőrincz, Z. Á. Milacski, B. Pintér, and A. L. Verő. Columnar Machine. *Biol. Insp. Cogn. Arch.*, 15:19–33, 2016.

[Lőrincz *et al.*, 2018] A. Lőrincz, M. Csákvári, Á. Fóthi, Z. Á. Milacski, A. Sárkány, and Z. Tősér. Towards reasoning based representations. *Cog. Sys. Res.*, 47:92–108, 2018.

[O'Donoghue *et al.*, 2016] B. O'Donoghue, E. Chu, N. Parikh, and S. Boyd. Conic optimization via operator splitting and homogeneous self-dual embedding. *J. Opt. Theo. App.*, 169(3):1042–1068, 2016.

[Pintér *et al.*, 2015] B. Pintér, Gy. Vörös, Z. Szabó, and A. Lőrincz. Wikifying novel words to mixtures of Wikipedia senses by structured sparse coding. In *Pattern Rec. App. Meth.*, pages 241–255. Springer, 2015.

[Regneri *et al.*, 2013] M. Regneri, M. Rohrbach, D. Wetzel, S. Thater, B. Schiele, and M. Pinkal. Grounding action descriptions in videos. *TACL*, 1:25–36, 2013.

[Romera-Paredes and Torr, 2015] B. Romera-Paredes and P. Torr. An embarrassingly simple approach to zero-shot learning. In *ICML*, pages 2152–2161, 2015.

[Schmidhuber, 2015] J. Schmidhuber. Deep learning in neural networks. *Neural Networks*, 61:85–117, 2015.

[Simon *et al.*, 2017] T. Simon, H. Joo, I. Matthews, and Y. Sheikh. Hand keypoint detection in single images using multiview bootstrapping. In *CVPR*, 2017.

[Sprechmann *et al.*, 2015] P. Sprechmann, A. M. Bronstein, and G. Sapiro. Learning efficient sparse and low rank models. *IEEE Tr. PAMI*, 37(9):1821–1833, 2015.

[Tibshirani *et al.*, 2005] R. Tibshirani, M. Saunders, S. Rosset, J. Zhu, and K. Knight. Sparsity and smoothness via the fused lasso. *J. Roy. Stat. Soc. B*, 67(1):91–108, 2005.

[Tősér and Lőrincz, 2015] Z. Tősér and A. Lőrincz. The cyber-physical system approach towards artificial general intelligence. In *Int. Conf. AGI*, pages 373–383. Springer, 2015.

[van der Maaten and Hinton, 2008] L. van der Maaten and G. Hinton. Visualizing data using t-SNE. *JMLR*, 9:2579–2605, 2008.

[Vijayanarasimhan *et al.*, 2017] S. Vijayanarasimhan, S. Ricco, C. Schmid, R. Sukthankar, and K. Fragkiadaki. SfMm-Net. *arXiv:1704.07804*, 2017.

[Wang *et al.*, 2007] Xuerui Wang, Andrew McCallum, and Xing Wei. Topical n-grams. In *Data Mining, IEEE ICDM*, pages 697–702. IEEE, 2007.

[Xian *et al.*, 2017] Yongqin Xian, Bernt Schiele, and Zeynep Akata. Zero-shot learning: the good, the bad and the ugly. In *CVPR*, pages 4582–4591, 2017.

Assisted and Incremental Medical Diagnosis using Explainable Artificial Intelligence

Isaac Monteath and Raymond Sheh
Intelligent Robots Group, Department of Computing
Curtin University, Bentley WA 6102, Australia
isaac.monteath@postgrad.curtin.edu.au, raymond.sheh@curtin.edu.au

***Abstract*—In this short paper, we introduce our preliminary work on leveraging Explainable Artificial Intelligence (XAI) principles, particularly as they relate to decision tree learning, to facilitate novel ways of providing decision support. We use the application of medical diagnosis, where a practitioner may have incomplete information and uses the AI system to both propose a diagnosis (class), as well as to suggest additional tests to yield attributes that may increase confidence in the diagnosis.**

XAI plays two vital roles in this application. First, the interaction between the (human) physician and the AI agent is collaborative; thus the physician needs to understand, at least to their satisfaction, the reasoning behind the agent's recommendations to effectively act on them. Systems that support XAI at the level of introspecting into the underlying decision-making process will tend to also use internal representations that support this kind of human-compatible understanding. Second, the healthcare field requires traceability and accountability; thus the use of XAI that is able to provide forensic justifications for decisions, especially when there are multiple stakeholders, is vital for widespread acceptance and regulatory compliance.

I. Introduction

Artificial Intelligence (AI) and related fields of Expert Systems and Decision Support Systems have been used in Medical Diagnosis for many years [1], [2]. In recent times, interest has increased as techniques have improved and as computation has become more cost-effective. Much of this progress has been in the field of Neural Networks, Deep Learning and associated techniques [3]. While these techniques are immensely powerful, their ability to model such complex relationships also hampers their ability to provide human-intelligible explanations about the way in which they make decisions, as well as the concepts that they have learned. Indeed, beyond issues of confidence, reliability and regulatory compliance, this obfuscation of the decision-making process by such AI techniques can also open the way to malicious activity [4].

In this paper, we present our work on a field of study called Explainable Artificial Intelligence (XAI) [5]. We discuss the opportunities that this presents for the use of AI in medical diagnosis. XAI is the study of AI techniques that can not only make decisions but can also provide explanations about those decisions and concepts that have been learned. We demonstrate a novel approach to performing disease classification in a medical application that includes the physician at every stage of the decision-making process. This approach involves making predictions incrementally using a decision tree learner, with the goal of classifying whether the patient has a particular condition. In this paper, we will use the Chronic Kidney Disease (CKD) dataset [6] as an example. At each stage of the learning process, the system will negotiate with the physician to decide on the next-best step to take towards a diagnosis.

We make use of decision trees [7] to model what our AI system learns. A decision tree learner is a machine learning classifier that represents its information in the form of a tree – a form of directed, acyclic graph. Each node in the tree represents a decision point whereby a logical test is applied to a single attribute. Execution flows down one branch (or, in probabilistic cases, through several) to the child nodes where another attribute test is applied until execution arrives at a leaf (or several leaves in the probabilistic case), which contains the class, or a probability distribution over classes, based on the training data. In the context of this medical application, "attribute" corresponds to the results of measurements and medical tests that relate to a patient, and from which we wish to make a diagnosis. For example, a Hemoglobin test yields an attribute that can be used for diagnosis. "Class" refers to a patient diagnosis, for example the identity, or presence or absence of, a disease.

The AI system builds the tree by repeatedly determining, for a given set of training examples, an attribute that most meaningfully separates those examples into two groups. For example, the presence or absence of high blood pressure in determining the likelihood of heart disease. The process then repeats separately for the two resulting groups.

On the one hand, this representation is less flexible than Neural Networks and, thus, the ability of the decision tree to learn complex relationships is not as great as a Neural Network. On the other hand, this structure also makes it possible to trace how decisions are made, in a way that can provide human-intelligible explanations. The partitioning of the decision-making process also means that the extent of errors caused by erroneous data can be clearly determined. Furthermore, the learning process is traceable and deterministic. As with many decisions to be made when producing a system, explainability and the ability to incrementally produce a decision in collaboration with a user needs to be traded off against other factors such as predictive accuracy and efficiency in an informed manner.

II. Explanation and Transparency

Explanations take many forms. The task of categorising explanations across various dimensions has been explored in cognitive science, psychology, and more recently in machine learning [8], [9], [10], [11]. We have selected the categorisation outlined in [12] as the most appropriate for the specific purpose of matching application requirements to specific capabilities of different AI techniques. In this context, we refer to explanations that come from the underlying decision-making process as "Introspective" explanations. This is in contrast to "Post-Hoc Rationalisations", which happen to match the data but are not derived from the underlying decision-making process. We also look at explanations that can trace a decision right back to the original historic data that the AI system's model learned from, a type of explanation that we refer to as "Model" explanation. Such explanations are vital in safety-critical systems, particularly where explanations are necessary for compliance, to determine errors and faults in the system, and to ensure that problems are fixed. Finally, we are interested both in explanations that "Justify" a given decision, as well as those that can "Teach" us something about the decision-making process.

In this paper we will focus on two types of explanation. The goal of these explanations is to allow the physician and the AI system to work together to come to a decision that is correct, transparent and understandable. The first type are those that assist in the process of incremental diagnosis while the second are those that trace the provenance of a decision.

A. Incremental Diagnosis

Incremental diagnosis involves an exchange between two agents. One agent, usually a human physician, has information about a specific case and seeks a diagnosis. The other agent could be an XAI agent (as in our case) or another human (perhaps a specialist) that has the historic knowledge to make, or assist in making, the required diagnosis. In the rest of this paper, we will refer to the former as the "physician" and the latter as the "diagnoser".

Incremental diagnosis involves the physician providing the diagnoser with partial information – some attributes, perhaps with some uncertainties. The diagnoser makes a preliminary diagnosis and also suggests what additional information, in the form of additional attributes, that would increase the certainty of the diagnosis. In the robotics and automation community, this can be considered a form of "active sensing", whereby the AI system doesn't just take the inputs that it is given, it also plays an active part in deciding, along with the physician, what information is sought.

An example is shown in Figure 1. In this simplified example, we have trained a decision tree learner on the "Chronic Kidney Disease" (CKD) dataset. We have modified the decision tree learner so that it is able to not only make decision, but it is also able to perform incremental diagnosis as described above. This technique should not be confused with "incremental learning", which refers to an on-line learning process in which the tree, or other learned model, may be updated when more training data becomes available [13], [14], [15]. By contrast,

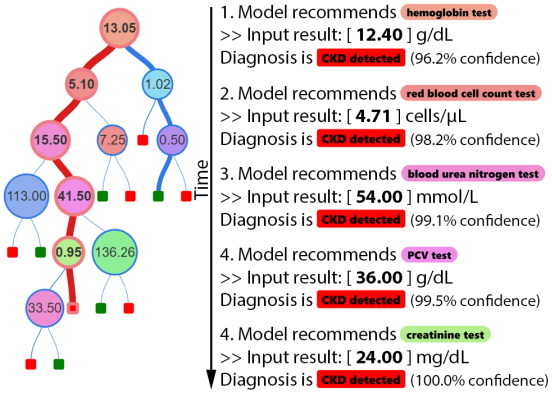


Fig. 1. An example of Incremental Diagnosis, using the "Chronic Kidney Disease" (CKD) dataset. The tree has been pruned for readability. Confidence percentages were generated based on the number of instances at each node.

our approach is referring to updating a prediction (using a decision tree learner) as more data about a *specific* instance becomes available.

For example, our system first recommends that, knowing nothing, the most informative test to perform is the *hemoglobin* test. The physician follows this recommendation and finds that the patient has a result of 12.40 g/dL. Through the decision tree, our AI system reports that there is a 96.2% confidence that CKD is present and that the result of the *red blood cell count* test will be the next most influential on this confidence.

This approach can also be used to determine, given an existing set of test results, which new tests will most improve the certainty of diagnosis. This is useful in situations where the physician may already have some test results, perhaps because they are easy to determine in the consulting room or perhaps because the patient already has them from a previous visit. Our AI system helps the physician to decide, for these given tests, which additional tests are worth ordering to best diagnose the patient and why. Figure 2 demonstrates how a proposed attribute at one iteration of the process can be visualised for the physician. Using this approach, the physician can quickly determine whether using a proposed attribute looks reasonable, as well as perform comparisons against other attributes they may have been considering.

It is important to note that our system is not just providing global importance metrics for the various tests. The recommendation for the next-most-informative test is given on the basis of the results of the preceding tests. This provides a very case-specific importance metric. In particular, a certain test may be generally uninformative except in a very small number of cases. Systems that make use of global dimensionality reduction will determine that, in general, this test is uninformative and ignore it completely. By contrast, our system will use previously-performed tests to determine whether a case is likely to benefit from the test in question, and use it if necessary.

Developing this importance metric is a key challenge that must be addressed, and is a subject of ongoing work. The metric will be used to determine what makes an attribute *important* for diagnosing the patient. This is critical because

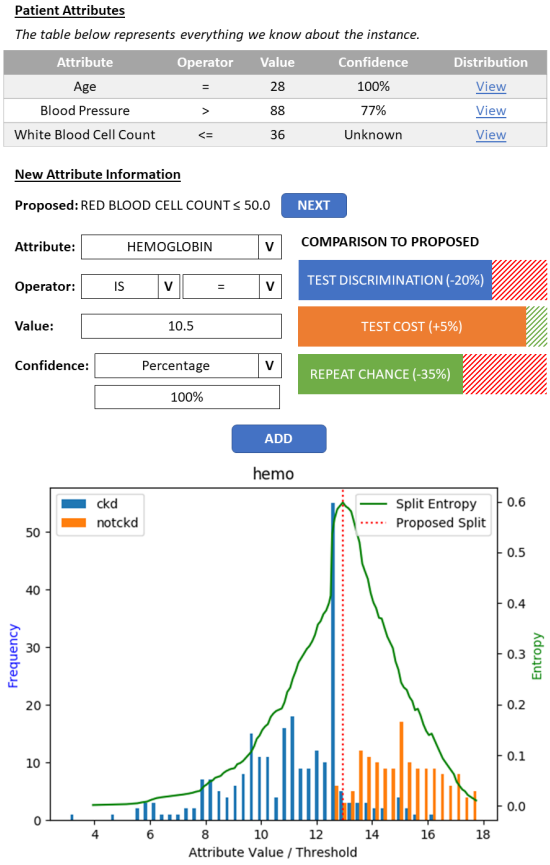


Fig. 2. Demonstration user interface which allows the user to input more information about the patient and interact with the decision tree. A histogram diagram is used to visualise the training data present at the current depth of the tree. The x-axis shows the possible values of the proposed attribute, *hemoglobin*. The green line shows the entropy computed by the tree for each point if it were used as the splitting threshold. The red dotted-line shows the threshold that is proposed by the system. Note that the histogram is generated from a real tree learned on the CKD dataset, however the comparison statistics and other values are placeholders for demonstration purposes.

it determines the order in which attribute tests are proposed to the user. In the context of XAI, this metric can be used to form an explanation that justifies why a particular attribute was chosen over another.

Perhaps the simplest metric is the height of the attribute in the tree. The tree is built by recursively examining attributes and determining the decision point, or split, that is most informative given the training data at that point. Those appearing higher up in the tree will tend to be more important given the training data available at build-time (the pruning process means that this will not always be strictly true). This is used in our initial implementation and demonstrated in Figure 1.

However, at run-time, the importance of a given split can depend on other factors that were not properly accounted for in these build-time entropy measures. For example, at run-time, classes might be grouped together (e.g. we only wish to tell time-critical from non-time-critical medical conditions) or we may have some prior belief (e.g. other factors that could not have been known when the tree was built might mean that certain classes should be excluded). Furthermore, there may be several potential attributes and splits that could have yielded similar information gains for a given node in the tree; distinguishing these cases from those where there was a clear most-informative attribute and split is significant from the perspective of informing the physician as to the importance of a given medical test. In this paper, we propose and describe alternative importance metrics that can handle such cases.

The usefulness of an attribute in classifying an instance is not the only factor used in making decisions. We propose to take into consideration other factors, such as the cost of performing the test (for example, in this application the monetary cost as well as the time-cost are significant factors).

In addition, our system will adapt to the capabilities of the physician by allowing a proposed test to be postponed or ignored completely. For example, the system may suggest that the most useful attribute is the one provided by the *hemoglobin* test but the physician may not be able to perform this test. The system enables a test to be skipped and will instead propose the next-most useful test. In another case, the physician may only have the resources to perform one test (or a select few). In this case, the physician can query the system to discover which of the tests they are able to perform will yield attributes that are most informative, given the other attributes about the patient that are already known to the system, and whether it is worth performing these tests at all. This negotiation between the system and physician can be captured and annotated by the physician to show a clear, explainable line-of-reasoning that led to a particular attribute being chosen for a split.

In some cases the physician may have a theory about the patient's diagnosis that they would like to test. This may not necessarily align with the tests proposed by the system. Even if the system proposes a better line of inquiry than the physician in the general case, the physician might have prior or external knowledge that justifies exploring their theory, or they may have slightly different motivations or requirements of the system than what it was designed or learned for – given the wide variety of patients it may not be possible to learn a meaningful model that accommodates all variations. While it is possible for the physician to simply override the system and run their own tests, this defeats the purpose of employing the system in the first place.

A useful compromise would be to allow the physician to input a prior distribution over classes. This indicates to the system which diagnoses they are interested or not interested in, and have the system adapt to account for this. While this prior cannot alter the structure of the tree, it can still be used to recompute the importance at each node and alter the recommendation. This information does not necessarily need to be provided at the beginning of the process; our system will be able to account for this dynamically. The physician's theories are likely to change as tests are completed, so they should be able to change which classes they are interested in at any point in time.

A related issue is that sometimes the classes used in the training data do not align with the physician's objectives at a given time. Sometimes the physician does not wish to make a distinction between a group of classes. For example, the first priority for the physician may be to determine whether the patient needs to be transferred to the emergency room.

Therefore, it is most important for the decision tree to prioritise tests that differentiate between time-critical and non-time-critical medical conditions. Differentiating between a large number of minor medical conditions is a lower priority than differentiating them from one major, time-critical one. The ability to provide class-likelihoods addresses this problem to an extent. However, in this case, the physician is only interested in two classes - critical and not-critical. A solution to this problem is to allow the physician to group classes, effectively indicating to the tree that all classes in a group are considered the same (for the purposes of the application).

We plan to integrate this functionality by re-computing the importance metric for a split after classes have been grouped and prior class weightings have been applied. Decision tree importance metrics (such as information gain) are generally based on the class distribution of training data at that node in the tree. Grouping classes is a simple operation that alters this distribution by merging the frequencies of grouped classes at the same point. The set of class-likelihoods is a distribution specified by the physician. By first applying the grouping transformation to the training data distribution and then multiplying it by the user-specified class-likelihood distribution, it is possible to calculate a new split-importance metric that takes the physician's knowledge into account. By analysing the information gain at each split, our approach also gives the user an indication of how sensitive their test needs to be before considering a different test.

B. Decision Tracing

In medical diagnosis, it is important to be able to trace the underlying source of the decision, right back to the data that was originally provided to the system when it was trained. There are three reasons for this.

First, it may be necessary to convince human supervisors, regulators, command hierarchy or some other entity that the decision is correct, or conversely to provide them with enough information to determine that the system has made an error, by explaining how the decision was made. The AI system is able to make decisions on the basis of experience in a number of cases significantly larger than any human is able to. This will inevitably cause disagreements between the AI system and any humans who may have a supervisory role over it. Being able to explain its decisions allows the humans to learn but also allows the system to be held accountable and encourages human engagement in the "checks and balances" process, ultimately improving patient outcomes.

A particular nuance that is important to note is that the explanation must be both convincing to human experts and be accurate in its reflection of the underlying decision-making process. Our approach satisfies these conditions. This is in contrast to "Post-Hoc Rationalisation" explanations [12]. Although these explanations may result in the same decision, they do not match the underlying decision-making process and may diverge in unpredictable ways from the AI system's predictions as attributes change. They are unlikely to satisfy regulatory requirements for transparency and are less useful for troubleshooting or making predictions about the behaviour of the system. This is often the case even if the system offers multiple explanations and asking the user to select between them [16] – it is by definition not possible to guarantee that the user has enough information to pick the actual, underlying explanation.

Second, if something goes wrong, it is necessary in such applications to determine where the error occurred and to apportion responsibility. This is necessary both for reasons of liability as well as for correcting the error. In such systems, the reason for an error may lie in user error, errors in data at run-time, a mismatch between the situation during training versus during deployment, an error in the training data itself or in bugs in the implementation [17]. Finally, if the AI system is in error, it is necessary to be able to verifiably determine the extent of the error and to determine the extent to which modifications to correct the system have affected its behaviour.

Our approach can determine not only how each attribute was used but also why it was used in that manner. This tracing can be performed right back to the underlying examples on which the model was trained. Our instrumented decision tree records, at each node, the statistics of the training examples that support that node's split and information to locate the underlying training examples. Pruning can obfuscate this process somewhat but it is still deterministic and the process can be expressed in relatively intelligible statistical terms. Furthermore, the way in which an optional prior and class groupings are incorporated into the decision process is also transparent right down to the original training data.

Our approach is effectively an incremental negotiation between two systems (one of which happens to be a human physician). This means that each step of the negotiation process can be logged and shown if required at a later time. At each stage of the process, the physician must decide on which new information to collect, and then record, a reason for their decision. The system acts as a guide to help the physician make a data-informed decision. In the case of a medical diagnosis, this is critical for tracing the reasoning behind the decision-making process.

III. Conclusion

Our XAI approach to incremental decision support for medical diagnosis allows AI systems to work alongside human experts, each informing the other and coming to a decision together. Our system is able to guide a physician in determining which test results are most useful given existing data. The system is also able to explain how a particular decision was made, tracing right back to the underlying training data. This provides the transparency that is crucial for patient confidence, regulation compliance, detecting and correcting errors, and improving patient outcomes.

The applications for this approach extend further than just medical applications. This approach is useful in any application that involves classifying an instance where data is gathered on that instance incrementally and on-demand. Our approach could be particularly useful in applications where gathering new information on an instance has a cost associated with it, such as time, money or bandwidth. As such, we plan to explore the use of our approach in applications of this nature.

REFERENCES

[1] R. O. Duda and E. H. Shortliffe, "Expert systems research," *Science*, vol. 220, no. 4594, pp. 261–268, 1983. [Online]. Available: https://saltworks.stanford.edu/assets/zm201ky2518.pdf

[2] J. A. Cruz and D. S. Wishart, "Applications of machine learning in cancer prediction and prognosis," *Cancer Informatics*, vol. 2, pp. 59–77, 2006.

[3] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[4] S. G. Finlayson, I. S. Kohane, and A. L. Beam, "Adversarial Attacks Against Medical Deep Learning Systems," pp. 1–16, 2018. [Online]. Available: https://arxiv.org/pdf/1804.05296.pdf

[5] D. Gunning, "Explainable Artificial Intelligence (XAI)," Defence Advanced Research Projects Agency (DARPA), Broad Agency Announcement DARPA-BAA-16-53, 2016.

[6] D. Dheeru and E. Karra Taniskidou, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[7] J. Quinlan, *C4.5: Programs for Machine Learning*, ser. Ebrary online. Elsevier Science, 2014. [Online]. Available: https://books.google.com/books?id=b3ujBQAAQBAJ

[8] H. Johnson and P. Johnson, "Explanation facilities and interactive systems," *Proceedings of the 1st international conference on Intelligent user interfaces - IUI '93*, no. July, pp. 159–166, 1993. [Online]. Available: http://portal.acm.org/citation.cfm?doid=169891.169951

[9] T. Miller, P. Howe, and L. Sonenberg, "Explainable AI: Beware of Inmates Running the Asylum," *IJCAI-17 Workshop on Explainable AI (XAI)*, pp. 36–42, 2017. [Online]. Available: https://arxiv.org/pdf/1712.00547

[10] F. Doshi-velez and B. Kim, "Towards A Rigorous Science of Interpretable Machine Learning," *arXiv preprint arXiv:1702.08608*, no. Ml, pp. 1–13, 2017.

[11] O. Biran and C. Cotton, "Explanation and Justification in Machine Learning : A Survey," *International Joint Conference on Artificial Intelligence Workshop on Explainable Artificial Intelligence (IJCAI-XAI)*, pp. 8–13, 2017.

[12] M. Roberts, I. Monteath, R. Sheh, D. W. Aha, P. Jampathom, K. Akins, E. Sydow, V. Shivashankar, and C. Sammut, "What was I planning to do?" *Proceedings of the ICAPS Workshop on Explainable AI Planning*, 2018.

[13] S. L. Crawford, "Extensions to the cart algorithm," *International Journal of Man-Machine Studies*, vol. 31, no. 2, pp. 197–217, 1989.

[14] P. E. Utgoff, "Incremental Induction of Decision Trees," *Machine Learning*, vol. 4, no. 2, pp. 161–186, 1989.

[15] P. Domingos and G. Hulten, "Mining high-speed data streams," *Kdd*, pp. 71–80, 2000. [Online]. Available: http://portal.acm.org/citation.cfm?doid=347090.347107

[16] A. S. Ross, M. C. Hughes, and F. Doshi-Velez, "Right for the right reasons: Training differentiable models by constraining their explanations," in *Proc. 26th Int'l Joint Conf. on AI (IJCAI-17)*, 2017.

[17] R. Sheh and I. Monteath, "Introspectively Assessing Failures through Explainable Artificial Intelligence," in *IROS Workshop on Introspective Methods for Reliable Autonomy*, 2017.

From Context Mediation to Declarative Values and Explainability

Grzegorz J. Nalepa[1], **Martijn van Otterlo**[2], **Szymon Bobek**[1], **Martin Atzmueller**[2]
[1] AGH University of Science and Technology
{gjn,szymon.bobek}@agh.edu.pl
[2] Department of Cognitive Science and Artificial Intelligence
Tilburg University, The Netherlands
{m.vanotterlo,m.atzmuller}@uvt.nl

Abstract

We argue that there are fruitful (inter)relations between value alignment of AI systems, trust in humans-AI interaction, and declarative representations that make AI systems transparent and explainable. We illustrate matters on two examples and end with directions for research.

1 Introduction

The recent rapid progress of AI enabled "intelligent" technologies to produce new, important and impressive applications. However, AI systems are increasingly covering new areas of decision making that were mostly reserved for humans so far. This raises a number of concerns and questions, i. e., what is a safe degree of autonomy of such systems, who would be *responsible* for their possible failure, or to what degree they can be *trusted*. In fact, there are even more serious, underlying, not so obvious issues, like to what degree do we actually *understand* how some of these systems work, or should they fail, can we explain why they failed, and how we can improve them once they fail. Another question could be whether the design procedure was maybe somehow flawed from the start? An even deeper question would be, which *values* guided the creators of these systems, and which values are actually *embedded* into the AI's code? Ultimately, for a truly general *adaptive* AI system, a crucial question is which values they have *obtained* over their lifetime. While such questions touch upon moral issues, they do relate to *engineering* as well since we *do* need to carefully *design* AI systems that make decisions which have (moral) consequences for people involved. Finally, an even more axiological question is about what should be the *underlying values* for the *creation* of AI systems which are safe, trustworthy, and possibly if not morally good, at least conforming to certain ethical norms.

Clearly, these issues are a current practical concern for engineering AI systems. Legal regulation is also slowly catching up with the rapid progress of AI. Prominent examples are the recent EU regulations on the explainability of AI systems, i. e., the new General Data Protection Regulation, and the "right to explanation", c. f., [Goodman and Flaxman, 2016]. Furthermore, several communities involved with AI systems have started to openly discuss these questions and started to identify *challenges* and *design principles*. The engineering community with the IEEE *Ethically Aligned Design* initiative [1], the robotics community with the EPSRC design principles for robotics systems [2], and the AI community at large with their own *Asilomar principles* [3] which explicitly talk about *judicial* and *failure* transparency: an AI system should be able to *explain* its decisions. All these efforts and concerns are rising regarding both the prospects [4] and benefits where the same concerns are raised about *malicious* [5] *use* of AI.

In our opinion, one of the most important of the underlying problems is the apparent lack of *transparency* of the decision making process in AI systems. From it stem several of the challenges outlined above, specifically regarding *understandability*, *explainability*, and the possibility of incremental improvement (after failures). Furthermore, we believe that this issue of transparency is mandatory to even start addressing the last group of questions, regarding the ethical dimensions of design and operation of intelligent systems.

With this background, in this short paper we assert that transparency in both design and operation of AI systems is needed for them to be explainable, safe, and trustworthy. If we want AI systems to gain a certain level of trust of humans, they need to operate according to human morals and values to some extent. Of course, this does not mean they need to obey some universal ethical system (e.g. Asimov's law's of robotics), but it does mean that it should be possible to demonstrate an alignment of their design and operation to certain ethical codes or sets of norms. Furthermore, we want to stress the role of *humans* in the design and improvement of operation of intelligent systems: *humans are responsible* for the design of AI systems. Moreover, if these systems are designed using machine learning techniques, people are responsible for *teaching* these systems. As we want AI systems to be adaptable and personalized to the needs and expectations of human operators and users, the teaching process does not end in a separate design phase. On the contrary, it continues during the operation and use of the system, and so does the responsibility of humans. To summarize, we claim not only

[1] https://ethicsinaction.ieee.org/
[2] https://epsrc.ukri.org/research/ourportfolio/themes/engineering/activities/principlesofrobotics/
[3] https://futureoflife.org/ai-principles/
[4] https://ai100.stanford.edu/
[5] https://maliciousaireport.com/

that humans have to be put in the loop of the design and use of AI systems. We claim, that humans never leave that loop. It is probably a good time for them to realize that.

In this paper, we illustrate our argument with examples that relate it to the design and operation of specific classes of AI systems and models they use. We believe that in order to achieve explainability we need to combine transparent symbolic models for reasoning with specific subsymbolic ones, such as probabilistic reasoning, and machine learning techniques. As such, AI systems have to be heterogeneous by design. Furthermore, we believe that putting the human user in the decision making loop of the system can contribute not only to the system performance, but can also improve the trust the user has in its operation. We claim that if appropriate conditions are met, users are willing to interactively improve how the system works for them. Finally, we demonstrate how specific transparent reasoning techniques can be used to explicitly model some ethical aspects of operation of the system.

2 Context

Transparency and explainability are vital for AI systems in two distinct ways: First, increasingly intelligent and pervasive systems introduced in our society need to be transparent and be able to explain their decisions. Second, explainability will also enhance trust at the user side, and due tothat improve the human-machine interaction performance. Our context is formed by three distinct research topics, discussed below.

Responsible AI The societal consequencs of AI beyond simple privacy and surveillance have become a hot topic in fields ranging from sociology to law under the name *ethics of algorithms* [van Otterlo, 2018b]. Algorithms basically transform *data* into *decisions*, where evidence can be inconclusive, inscrutable or misguided and this can cause many ethical consequences of actions, relating to *fairness*, *opacity*, *unjustified actions*, and *discrimination*. Overall, algorithms have an impact on *privacy* and can have *transformative effects* on *autonomy* [Mittelstadt *et al.*, 2016]. Transparency and the ability to *explain* AI decision making are core requirements for important aspects such as *trust*, *liability*, *responsibility* and *accountability* of algorithms [Diakopoulos, 2016]. In AI and machine learning itself, topics of interest are to ensure *fairness*, *accuracy*, *confidentially* and *transparency* (FACT). Especially transparency has been addressed by *explanation-aware computing* [Atzmueller and Roth-Berghofer, 2011]. Interestingly, the way humans have dealt with ethical issues among people can provide insights into how to deal with ethical consequences of AI. For example, human *codes of ethics* are based on principles of transparency and the ability to explain to the public what norms and values are in particular professions [van Otterlo, 2018a; van Otterlo, 2018b]. Efforts from researchers inside and outside AI enable to understand AI systems and their impact, thereby regulating the (legal) consequences of AI better, and as a result increase trust that the AI will "do the right thing".

Adaptive Human-Machine Interaction Interaction between humans and machines is studied in *human-robot interaction*, *human-computer interaction*, and *interactive machine learning* [Cuayáhuitl *et al.*,]. Until recently, often such interactions were studied with modified *reinforcement learning* [Wiering and van Otterlo, 2012], i. e., how to let the human actor provide guidance to the robotic learner. For example, humans can provide *rewards* or *provide answers to questions from the robot*, and these constructs can be embedded in formal models of goal-oriented tasks. Transparency of the robots behavior, or the interaction, was not an issue so far, although work in *relational robotics* has worked on learning *declarative, probabilistic skill representations* [Moldovan *et al.*, 2012]. Lately, focus has shifted (with influences from responsible AI) to the *value aligment* problem [Abel *et al.*, 2016; Taylor *et al.*, 2017], which is the challenge to construct (adaptive) systems that behave in such a way that they are *aligned* with human values. Narrow interpretations result in various kinds of *inverse* reinforcement learning, but many broader issues are studied to make machine learning systems "safe", including safe exploration, robust generalization over tasks, and avoiding negative side effects of truly intelligent AI which may alter their own reward function at will.

Computing Explanations Recently, the concept of transparent and explainable models has gained a strong focus and momentum in the machine learning and data mining community. Several methods focus on specific model types, e. g., tree-based models [Tolomei *et al.*, 2017]. Also, methods for associative classification, e. g., class association rules [Atzmueller *et al.*, 2018] can be applied for obtaining explicative, i. e., transparent, interpretable, and explainable models [Atzmueller, 2017]. Then, individual steps of a decision can be traced-back to the model, similar to *reconstructive explanations*, c. f., [Wick and Thompson, 1992] on several explanation dimensions [Atzmueller and Roth-Berghofer, 2011]. While the methods sketched above focus on specific modeling methods, there are several approaches for model agnostic explanation methods, e. g., [Ribeiro *et al.*, 2018]. General directions are given by methods considering counterfactual explanation, e. g., [Mandel, 2007; Wachter *et al.*, 2017]. Furthermore, other general methods consider data perturbation and randomization techniques as well as interaction analysis methods, e. g., [Henelius *et al.*, 2017]. Deep neural network models have become a default learning paradigm for huge amounts of data in computer vision, linguistics and reinforcement learning. Their black-box nature has recently triggered several strands of research e.g. focusing on *distilling* learned knowledge into transparent representations such as trees [Frosst and Hinton, 2017] or computing explanations of policy behavior using object saliency maps [Iyer *et al.*, 2018].

Logical, declarative representations, by their nature, provide more options for interpretation [Srinivasan, 2001] and explanation-based reasoning [Atzmueller and Seipel, 2008; van Otterlo, 2009]. With *probabilistic programming languages* these can also be used effectively for machine learning [De Raedt, 2008], e. g., *relational* reinforcement learning [van Otterlo, 2012]. *Comprehensible* machine learning systems can be very useful for transparent and explainable AI.

In the next two sections we will briefly relate to our recent work addressing the selected problems emphasized in the introduction. The first case concerns systems that combine symbolic reasoning with uncertainty handling mechanism supported by the user in an interactive manner.

3 Context Mediation with Human in the Loop

Intelligibility in context-aware systems is an ability of the system to being understood by its users [Lim *et al.*, 2009]. In human-centric systems, such as cognitive advisors this feature is of a key importance, as one of their primary goals is to build trust with a user that helps getting deeper insight into personal preferences and habits. This trust can be built in many different ways, one of which is keeping human in the loop of the decision and possibly learning process. In our recent work, this was achieved by providing an implicit mediation mechanism proposed in [Bobek and Nalepa, 2017a].

The key idea behind mediation is to involve the user into the decision making process, in cases where the output of the inference may be uncertain, and thus may reduce user's trust. In Fig. 1, the architecture of the semantic mediation system for implicit user feedback is shown. The system consists of two main parts: the static knowledge component and the dynamic knowledge component. The static knowledge component provides a semantic representation of the user environment. It allows for communication with the user with the use of concepts that are easily understandable for him. It is an input for the dynamic knowledge component which is responsible for question generation, where questions use concepts from the ontology. These questions aim at obtaining additional information from the user, which can be used to resolve ambiguities or to create new knowledge. Furthermore, the dynamic knowledge component allows for an online mediation between the user and the system.

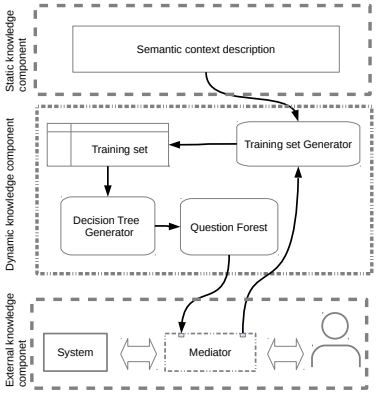


Figure 1: Architecture of the implicit semantic mediation system

The system was used by us in a dead-reckoning localization system [Köping *et al.*, 2015], where the implicit mediation was triggered, and where the localization module was highly uncertain about the user position. Based on the ontology that described the environment, it generated easy to answer questions about the most probable user locations. On the one hand this allowed the system to resolve uncertainty, but on the other hand it gave the user better insight on how the decision about his or her location was made. The system we used for localisation was a particle filtering method, a purely mathematical, probabilistic model, and hardly interpretable. We combined it with our implicit mediation system to support the particle filtering algorithm in cases where the estimates of the localisation was very unclear. The localisation of the user was estimated based on the clusters generated from particles. If more than one cluster pointed to different locations on the map, the mediation was triggered.

Uncertainty of knowledge was an inevitable and important challange in the mediation process. It stemed from the fact that sometimes there was no knowledge about the environment (possible location of the user were in rooms which we did not have map of). Furthermore, the user might not be sure about his or her answers. To address this challange, we used a *rule-based* approach combined with *certainty-factors* to allow both uncertainty due to the incompleteness of the model and uncertainty of user answers [Bobek and Nalepa, 2017b].

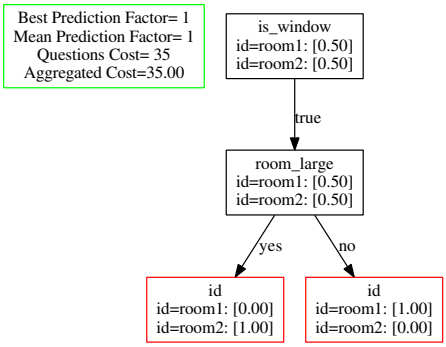


Figure 2: Example of a decision tree from a question forest

The example in Fig. 2 shows the decision tree chosen by the mediation system from the question forest generated for two ambiguous locations. The tree can be then translated into rules of the form presented in Listing 1. Note that the classification accuracy is translated into the form of a certainty factor preceded by the # operator. It allows the user to answer a yes-or-no question with uncertainty as well, and incorporates this uncertainty into reasoning process. Thus, the final result from the reasoning process is also a probabilistic value that can be used by particle filtering algorithm, or with any other subsymbolic method. This extended rule notation is handled by a dedicated inference engine – HeaRTDroid [Bobek and Nalepa, 2017b]. It allows for handling rule uncertainty, but also to operate on attributes related to statistical features of sensor data to deal with machine imprecision.

```
xrule mediate/1: [is_window eq yes,room_large eq yes] ==>
     [id set room2] # 1.0
xrule mediate/1: [is_window eq yes,room_large eq no ]  ==>
      [id set room1] #.1.0
xrule mediate/1: [is_window eq no]==> [id set room1] #-1.0
xrule mediate/1: [is_window eq no]==> [id set room2] #-1.0
```

Listing 1: HMR+ rules representing question tree

We move on to our second case leveraging the combination of declarative models with probabilistic reasoning even more.

4 Declarative Ethical Programs

AI programs should be able to optimize and explain behavior. *Declarative decision-theoretic ethical programs* (DDTEPs) [van Otterlo, 2018a] declaratively specify (and solve) *decision-theoretic problems*, based on the probabilistic programming language (PLL) DT-PROBLOG [Van den Broeck *et al.*, 2010]. Solutions are computed by considering all *possible worlds* modeled by the program. The general idea is to formalize what is known explicitly in the model,

and use *reasoning* to compute optimal decisions. DDTEPs fit into logical approaches for ethical (or: value-driven) reasoning [Anderson and Anderson, 2007] but also *relational reinforcement learning* [van Otterlo, 2012] and provides opportunities for *explanation-focused* computations [van Otterlo, 2009] by reasoning over the logical parts of the model.

A (partial) toy example of a DDTEP for a **self-driving car** consists of a decision to either `run_into_wall` (killing the passenger) or a `collision` (killing a pedestrian). We can specify *percepts* for what is in front of the car and a rule that says what happens when a collision is made. The `utility` function defines the *value* of each outcome, making the optimal value -30 (amounting to kill the passenger by steering away).

```
(action)   ?::run_into_wall; ?::collision.
(percepts) in_front_of_car(a).  baby(a).
           in_front_of_car(b). pedestrian(b). ...
(rules)    kill(X) :- in_front_of_car(X), collision.
(values)   utility(run_into_wall, -30).
           utility(kill(X), -20) :- pedestrian(X).
           utility(kill(X), -40) :- baby(X).
```

DDTEPs proved successful for toy ethical domains, e. g., *cake-or-die* and *burning room* [Abel *et al.*, 2016]. There, an agent does not know all values and norms but it can *ask*. This effectively renders the decision problem *partially observable*, requiring *information gathering* first. Through this "human-in-the-loop" (and value alignment), the AI is told explicitly what the norms are. Specifying *human* values for a DDTEP comes with choices, such as that a `baby` is worth more than a `pedestrian` (or even that they appear on the same *scale*). Such choices can be dependent on social, cultural and other factors as the large-scale experiment *the Moral Machine* [6] aims to investigate, in which people are confronted with ethical *dilemmas* (for autonomous cars) to reveal their intrinsic values.

PLLs typically support *learning*, such that rule parameters (e.g. relative values) can be learned from data instead. Let us take a **fair access in archives** case [van Otterlo, 2018b] where a decision must be made whom to supply with newly disclosed archival material, for example using estimates of a researcher's `authority`. Then, we make the rules *probabilistic*.

```
(rules)  0.1::reach(X):-person(X),social_network(X,small).
         0.9::authority(X):-person(X),h_index(X,high).
     impact(P,T):-topic(T),authority(P).
(action) ?::give(P,T):-person(P),topic(T).
(value)  score(P,T):-give(P,T),impact(P,T).
         utility(score(P,area51),100):-person(P).
```

In this (partial) example, the decision logic dictates that impact depends on authority, which is probabilistically dependent on h-index. The logic is transparent, whereas the numbers can vary with incoming data.

DDTEP open up the black box of algorithms and make decision logic transparent. Still they also allow for machine learning to fill in additional details from data. This general pattern is a solution to value alignment in AI systems in complex domains: i) formalize existing norms and values transparently into a DDTEP, and ii) finetune parts of the program on data. Finetuning is needed because norms and values are never complete and domains are inherently stochastic. Logical formalism such as DDTEP also support *reasoning* about sequential processes, including *explanation-based* reasoning about satisfying norms or obtaining values, opening up the possibility to *ethically explain* its behavior.

[6] http://moralmachine.mit.edu/

5 Outlook and Conclusions

We have argued and illustrated that both declarative representations and the interaction between AI and humans are important factors in getting transparent and explainable systems, also incorporating ideas of *explanation-aware computing* [Atzmueller and Roth-Berghofer, 2011; **?**]. In the first example it was shown that the combination of an explainable system setup and the opportunity for the user to inspect knowledge and reasoning of the system and to provide feedback, was good for both the trust of the user as well as performance of the system. The second example showed how declarative representations can be used to open up the reasoning of systems that need to make (ethical) decisions and to explain how they get to them. This way again the user can be more engaged in the interaction with the AI and thus more opportunities rise to obtain (optimal) value alignment. Because that is what both examples show: explainability by transparency combined with human users will hopefully result in humans and AI agree more and more on "what is the right thing to do for the AI". Such explainable constructs should be used in the design phase, the interaction (use) phase, but definitely also in the improvement phase of the system over time. As a conclusion, we argue that in order for AI systems to be explainable, they need to be set up as hybrid systems with a good use of declarative knowledge from the start. Explainable systems, we conjecture, will be more trusted by humans and that will increase both performance and value alignment. For achieving these goals, much more research is needed; we want to mention four core directions.

1: Hybrid frameworks that combine declarative knowledge and symbolic reasoning with machine learning. Today AI systems grew far to complex for a single class of models to be sufficient to address problems these systems are supposed to solve. However, there is still an apparent lack of appropriate design approaches for hybrid systems.

2: Integrate explanation-based approaches to reason, provide feedback and support failure analysis. Recent works on explanation generation need to be combined with classic AI approaches to *abductive reasoning*, and *diagnosis*. These mechanisms should not only be used for explanation, but more importantly for provisioning of hypotheses how to ameliorate the system in an understandable way.

3: Use formal analysis techniques to qualify and quantify the performance of the system in terms of value alignment. An important aspect of hybrid frameworks concerns the *formal verification* of their properties on the possibly ethical level, e.g. by expressing all in a (decision-theoretic) logic, it becomes possible to *prove* properties or to analyze *executable ethical specifications* by looking at their potential errors or ethical-logical inconsistencies, or even quantify the "level"of value alignment achieved so far.

4: Interdisciplinary studies where the performance is analyzed in terms of trust, value alignment, and ethical codes. Clearly AI engineers need to work in integrated yet inter/multidisciplinary teams with knowledgeable researchers from applied ethics, law and sociology to address the issues we outlined. *This is urgent, as AI has already became an important part of our lives, and will continue to be even more so.*

References

[Abel *et al.*, 2016] D. Abel, J. MacGlashan, and M.L. Littman. Reinforcement Learning as a Framework for Ethical Decision Making. In *AAAI Workshop: AI, Ethics, and Society*, 2016.

[Anderson and Anderson, 2007] M. Anderson and S.L. Anderson. Machine Ethics: Creating an Ethical Intelligent Agent. *AI Magazine*, 28:15–26, 2007.

[Atzmueller and Roth-Berghofer, 2011] M. Atzmueller and T. Roth-Berghofer. The Mining and Analysis Continuum of Explaining Uncovered. In *Research and Development in Intelligent Systems XXVII*, pages 273–278. Springer, 2011.

[Atzmueller and Seipel, 2008] Martin Atzmueller and Dietmar Seipel. Declarative Specification of Ontological Domain Knowledge for Descriptive Data Mining. In *Proc. International Conference on Applications of Declarative Programming and Knowledge Management (INAP)*. Springer, 2008.

[Atzmueller *et al.*, 2018] M. Atzmueller, N. Hayat, M. Trojahn, and D. Kroll. Explicative Human Activity Recognition using Adaptive Association Rule-Based Classification. In *Proc. IEEE International Conference on Future IoT Technologies*. IEEE, 2018.

[Atzmueller, 2017] M. Atzmueller. Onto Explicative Data Mining: Exploratory, Interpretable and Explainable Analysis. In *Proc. Dutch-Belgian Database Day*. TU Eindhoven, Netherlands, 2017.

[Bobek and Nalepa, 2017a] S. Bobek and G. J. Nalepa. Uncertain Context Data Management in Dynamic Mobile Environments. *Future Generation Computer Systems*, 66(January):110–124, 2017.

[Bobek and Nalepa, 2017b] S. Bobek and G. J. Nalepa. Uncertainty Handling in Rule-based Mobile Context-Aware Systems. *Pervasive and Mobile Computing*, 39(August):159–179, 2017.

[Cuayáhuitl *et al.*,] H. Cuayáhuitl, N. Dethlefs, L. Frommberger, M. Van Otterlo, and O. Pietquin, editors. *Proceedings of Machine Learning Research*, volume 43. PMLR.

[De Raedt, 2008] L. De Raedt. *Logical and Relational Learning*. Springer, 2008.

[Diakopoulos, 2016] N. Diakopoulos. Accountability in Algorithmic Decision Making. *Communications of the ACM*, 59(2):56–62, 2016.

[Frosst and Hinton, 2017] N. Frosst and G. E. Hinton. Distilling a Neural Network Into a Soft Decision Tree. *CoRR*, abs/1711.09784, 2017.

[Goodman and Flaxman, 2016] B. Goodman and S. Flaxman. European Union Regulations On Algorithmic Decision-Making and a "Right to Explanation". *arXiv preprint arXiv:1606.08813*, 2016.

[Henelius *et al.*, 2017] Andreas Henelius, Kai Puolamäki, and Antti Ukkonen. Interpreting Classifiers through Attribute Interactions in Datasets. *Proc. 2017 ICML Workshop on Human Interpretability in Machine Learning (WHI 2017)*, 2017.

[Iyer *et al.*, 2018] R. Iyer, Y. Li, H. Li, M. Lewis, R. Sundar, and K. Sycara. Transparency and Explanation in Deep Reinforcement Learning Neural Networks. In *Proceedings of the AAAI/ACM Conference on Artificial Intelligence, Ethics, and Society*, 2018.

[Köping *et al.*, 2015] L. Köping, M. Grzegorzek, F. Deinzer, S. Bobek, M. Ślażyński, and G. J. Nalepa. Improving indoor localization by user feedback. In *Proc. International Conference on Information Fusion*, pages 1053–1060, July 2015.

[Lim *et al.*, 2009] B. Y. Lim, A. K. Dey, and D. Avrahami. Why and Why Not Explanations Improve the Intelligibility of Context-aware Intelligent Systems. In *Proc. SIGCHI*, CHI '09, pages 2119–2128, New York, NY, USA, 2009. ACM.

[Mandel, 2007] D. R Mandel. Counterfactual and Causal Explanation: From Early Theoretical Views To New Frontiers. In *The Psychology of Counterfactual Thinking*, pages 23–39. Routledge, 2007.

[Mittelstadt *et al.*, 2016] B.D. Mittelstadt, P. Allo, M. Taddeo, S. Wachter, and L. Floridi. The Ethics of Algorithms: Mapping the Debate. *Big Data & Society*, 3(2), 2016.

[Moldovan *et al.*, 2012] B. Moldovan, P. Moreno, M. van Otterlo, J. Santos-Victor, and L. De Raedt. Learning Relational Affordance Models for Robots in Multi-Object Manipulation Tasks. In *Proc. ICRA*, 2012.

[Ribeiro *et al.*, 2016] M. T. Ribeiro, S. Singh, and C. Guestrin. Model-Agnostic Interpretability of Machine Learning. *ICML Workshop on Human Interpretability in Machine Learning*, 2016.

[Ribeiro *et al.*, 2018] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Anchors: High-Precision Model-Agnostic Explanations. AAAI, 2018.

[Srinivasan, 2001] A. Srinivasan. Four Suggestions and a Rule Concerning the Application of ILP. In S. Dzeroski and N. Lavrac, editors, *Relational Data Mining*, chapter 15, pages 365–374. Springer, 2001.

[Taylor *et al.*, 2017] J. Taylor, E. Yudkowsky, P. LaVictoire, and A. Critch. Alignment for Advanced Machine Learning Systems, 2017. MIRI (unpublished) https://intelligence.org/2016/07/27/alignment-machine-learning/.

[Tolomei *et al.*, 2017] G. Tolomei, F. Silvestri, A. Haines, and M. Lalmas. Interpretable Predictions of Tree-based Ensembles via Actionable Feature Tweaking. In *Proc. KDD*. ACM, 2017.

[Van den Broeck *et al.*, 2010] G. Van den Broeck, I. Thon, M. Van Otterlo, and L. De Raedt. DTProbLog: A Decision-Theoretic Probabilistic Prolog. In *Proceedings of AAAI*, 2010.

[van Otterlo, 2009] M. van Otterlo. Intensional Dynamic Programming: A Rosetta Stone for Structured Dynamic Programming. *Journal of Algorithms*, 64:169–191, 2009.

[van Otterlo, 2012] M. van Otterlo. Solving Relational and First-Order Markov Decision Processes: A Survey. In M.A. Wiering and M. van Otterlo, editors, *Reinforcement Learning: State-of-the-art*, chapter 8, pages 253–292. Springer, 2012.

[van Otterlo, 2018a] M. van Otterlo. From Algorithmic Black Boxes to Adaptive White Boxes: Declarative Decision-Theoretic Ethical Programs as Codes of Ethics. In *Proc. of the AAAI/ACM Conference on Artificial Intelligence, Ethics, and Society*, 2018.

[van Otterlo, 2018b] M. van Otterlo. Gatekeeping Algorithms with Human Ethical Bias: The Ethics of Algorithms in Archives, Libraries and Society, 2018. https://arxiv.org/abs/1801.01705.

[Wachter *et al.*, 2017] S. Wachter, B. Mittelstadt, and C. Russell. Counterfactual Explanations without Opening the Black Box: Automated Decisions and the GDPR. 2017.

[Wick and Thompson, 1992] M. R. Wick and W. B. Thompson. Reconstructive Expert System Explanation. *Artif. Intell.*, 54(1-2):33–70, 1992.

[Wiering and van Otterlo, 2012] M.A. Wiering and M. van Otterlo, editors. *Reinforcement Learning: State-of-the-art*. Springer, 2012.

Local Interpretable Model-agnostic Explanations of Bayesian Predictive Models via Kullback–Leibler Projections

Tomi Peltola[1]
[1] Helsinki Institute for Information Technology HIIT, Department of Computer Science,
Aalto University, Helsinki, Finland
tomi.peltola@aalto.fi

Abstract

We introduce a method, KL-LIME, for explaining predictions of Bayesian predictive models by projecting the information in the predictive distribution locally to a simpler, interpretable explanation model. The proposed approach combines the recent Local Interpretable Model-agnostic Explanations (LIME) method with ideas from Bayesian projection predictive variable selection methods. The information theoretic basis helps in navigating the trade-off between explanation fidelity and complexity. We demonstrate the method in explaining MNIST digit classifications made by a Bayesian deep convolutional neural network.

1 Introduction

Interpreting why a machine learning model makes a certain prediction for given inputs can be difficult for complex models, but is important for evaluating the trustworthiness of the predictions and for model criticism and development. Explaining a prediction requires understanding the proper context of the prediction (inputs and what kind of changes the inputs could meaningfully have) and how the prediction would react to changes in the context (the mapping from inputs to the prediction). Understanding non-linear predictive models globally would require understanding the underlying prediction function at all locations of the input space. Local interpretations are more feasible and many methods have been proposed for this (e.g., [Baehrens *et al.*, 2010; Ribeiro *et al.*, 2016; Lundberg and Lee, 2017]). They rely on the assumption that non-linear functions can be approximated by simpler, more interpretable functions locally.

In addition to interpretability, properly accounting for uncertainty in the predictions is important in many applications, for example, in medicine. In Bayesian probabilistic modelling, the posterior distribution of the model parameters (the conditional distribution induced by the modelling assumptions and training data) and posterior predictive distributions naturally capture the uncertainty about the parameters and model structure (epistemic uncertainty) in addition to noise in prediction (aleatoric uncertainty). When explaining predictions, these uncertainties should also be accounted for.

We introduce a method for Kullback–Leibler divergence based local interpretable model-agnostic explanations, KL-LIME, that extends the recently proposed local interpretation method LIME [Ribeiro *et al.*, 2016] to Bayesian models (although it can also be used for non-Bayesian probabilistic models) and provides a principled way to handle different types of predictions (continuous valued, class labels, counts, censored and truncated data, etc.). The proposed method is based on combining the LIME approach with methods introduced for variable selection in Bayesian linear regression models [Goutis and Robert, 1998; Dupuis and Robert, 2003; Peltola *et al.*, 2014; Piironen and Vehtari, 2017]. The method works by fitting an interpretable explanatory model (e.g., a sparse linear model) to locally match the original prediction model via minimizing the Kullback–Leibler divergence between their predictive distributions.

The next section gives a more detailed background on LIME and Bayesian projection predictive variable selection. Section 3 describes the proposed method, KL-LIME, and Section 4 demonstrates it in explaining deep convolutional neural network predictions. Section 5 presents a summary and conclusions.

2 Background

In this section, we describe the LIME method and projection predictive variable selection. In the next section, they are combined to extend LIME to Bayesian predictive models.

2.1 Local Interpretable Model-agnostic Explanations – LIME

The Local Interpretable Model-agnostic Explanations (LIME) method of Ribeiro et al. [2016] provides local explanations of predictions of a classifier f by fitting a simpler, interpretable explanation model g locally around the data point x of which classification is to be explained. The explanation model is fit on an interpretable representation of the original data space. For example, let $x \in \mathbb{R}^d$ be a vector of the gray scale values of pixels in an image. An interpretable representation $x' \in \{0,1\}^{d'}$ might then be a vector of binary values representing the absence or presence of pixels in the image (absence meaning having the value of a background color, e.g., white). The LIME explanation $\hat{g}$

arises by solving the optimization problem

$$\hat{g} = \arg\min_{g \in G} \mathcal{L}(f, g, \pi_x) + \Omega(g),$$

where G is the explanation model family, $\mathcal{L}$ is a loss function, π_x defines the locality around x, and Ω is a complexity penalty.

In practice, G is taken to be the set of linear regression models, with Ω restricting that only some number of the explanatory features can have non-zero regression weights (although other types of explanation models could be used). The loss function is taken to be the weighted L2 distance

$$\mathcal{L}(f, g, \pi_x) = \sum_i \pi_x(z_i)(f(z_i) - g(z_i'))^2,$$

where the sum goes over a set of sampled perturbed points around x, $\{(z_i, z_i'), i = 1, \ldots, m\}$, where z_i is a perturbed data point in the original data space and z_i' the corresponding interpretable representation. $\pi_x(z_i)$ weights the samples based on their similarity to x, the point where the classification result is being explained.

2.2 Projection predictive variable selection

Variable selection is the problem of choosing a smaller set of covariates from among the full set available and is often used to simplify high-dimensional regression models. In Bayesian modelling, sparsity-inducing and shrinkage priors can be used to regularize high-dimensional regression models. However, they do not lead to truly sparse posterior distributions, since, with finite data, there will remain uncertainty about whether some covariates should or should not be included in the regression. Projection predictive variable selection is an approach to variable selection in Bayesian regression models, which removes covariates that do not considerably contribute to the explanatory power of the full model [Goutis and Robert, 1998; Dupuis and Robert, 2003; Peltola *et al.*, 2014; Piironen and Vehtari, 2017]. It works by projecting the information in the model encompassing all variables to a model that uses only a subset of the variables and has been empirically shown to have competitive performance compared to other variable selection approaches [Piironen and Vehtari, 2017].

For a prediction model M, defined on the full set of covariates, let $p(y \mid x, \theta, M)$ be the observation model of the target variable y given covariates x and model parameters θ and $p(\theta \mid D, M)$ the posterior distribution of the parameters given a training dataset D. Given the parameter θ of the full model M, the projection predictive variable selection approach fits a model M_s with a subset s of the covariates, denoted by x_s with $x = (x_s, x_{\setminus s})$, by minimizing the Kullback–Leibler divergence to the full model M

$$\hat{\theta}_s = \arg\min_{\theta_s} \frac{1}{N} \sum_{i=1}^{N} \mathrm{KL}[p(y \mid x_i, \theta, M) \| p(y \mid x_{i,s}, \theta_s, M_s)],$$

where i runs over the N training samples in the dataset D. In practice, the optimization is solved L times for L samples $\theta^{(l)}, l = 1, \ldots, L$, from the posterior distribution $p(\theta \mid D, M)$ to get a projected posterior distribution for the model M_s. The total information loss δ of using the subset x_s instead of full set x is then approximated as the average of the above loss over the samples from the posterior distribution:

$$\delta[M \| M_s] \approx \frac{1}{LN} \sum_{l=1}^{L} \sum_{i=1}^{N} \mathrm{KL}[p(y \mid x_i, \theta^{(l)}, M) \| p(y \mid x_{i,s}, \hat{\theta}_s^{(l)}, M_s)].$$

For generalized linear models, the optimization problems are related to the generalized linear model estimation equations [Goutis and Robert, 1998]. The projection method is, however, not limited to linear regression. For example, the approach has also been used for variable selection in Gaussian processes [Piironen and Vehtari, 2016]. A similar KL divergence minimization approach, with a further regularization penalty, is used in [Tran *et al.*, 2012] to define a predictive lasso method.

For variable selection, the projections are supplanted with a search process for finding a good subset of the covariates. This needs to weigh the benefits and costs of keeping a number of covariates. Dupuis and Robert [2003] introduced the relative explanatory power

$$1 - \frac{\delta[M \| M_s]}{\delta[M \| M_0]}, \tag{1}$$

where M_0 is a null model (e.g., the model without any covariates), to quantify the quality of the subset models and to help in determining the best subset model by choosing the smallest model that retains enough of the explanatory power of the full model. An alternative approach was suggested in [Peltola *et al.*, 2014], where cross-validation was combined with the projection predictive approach to estimate out-of-sample prediction performances at each model size along a forward selection path.

3 KL-LIME for explaining predictions of Bayesian models

We combine ideas from LIME and projection predictive variable selection to define the KL-LIME method for Kullback–Leibler divergence based local interpretable model-agnostic explanations of Bayesian predictive models (although the method can also be applied on non-Bayesian probabilistic models). Let $p(y \mid x, \theta, M)$ be the observation model of the predictive model M given input x and $p(\theta \mid D, M)$ the posterior distribution of its parameters θ given a dataset D. Similar to LIME, we define the explanation as an interpretable model $p(y \mid x', \phi, G)$ from (now a probabilistic) model family G with parameters ϕ, and possibly operating on a simplified representation x' of the original input x. Similar to projection predictive variable selection, the parameters of the explanation model are found by minimizing its Kullback–Leibler

divergence from the predictive model M:

$$\begin{aligned}\hat{\phi}^{(l)} = \arg\min_{\phi} & \int \pi_x(z)\,\mathrm{KL}[p(y \mid z, \theta^{(l)}, M) \| p(y \mid z', \phi, G)]dz \\ & + \Omega(\phi) \\ \approx \arg\min_{\phi} & \frac{1}{N}\sum_{i=1}^{N} \mathrm{KL}[p(y \mid z_i, \theta^{(l)}, M) \| p(y \mid z_i', \phi, G)] \\ & + \Omega(\phi),\end{aligned}$$

for $l = 1, \ldots, L$ posterior samples from $p(\theta \mid D, M)$ and where $\pi_x(z)$ is a probability distribution (we assume there is a mapping between z to z') defining the local input data space neighborhood around x, the data point which prediction is to be explained. Ω penalizes the complexity of the explanation model. In practice, the expectation over locality π_x is computed via a Monte Carlo approximation by sampling N points from $\pi_x(z)$.

To measure the fidelity of the explanation, we can compute the relative explanatory power between M and G, following Equation 1, with an appropriate definition of the null model. Alternatively, one could also compute more direct measures of the performance, such as mean squared error or classification accuracy by sampling a test set from the locality distribution $\pi_x(z)$.

In the demonstration of the approach below, we use linear models for the explanation model family and L1 regularization for the complexity penalty Ω. The KL minimization can then be solved with lasso regression (or generalized linear model variants of lasso regression). For non-Bayesian probabilistic models, one would not have posterior samples, but only a point estimate of the parameters θ which can be projected to the explanation parameters ϕ.

4 Demonstration

We demonstrate the proposed method in explaining deep convolutional neural network predictions in the MNIST dataset of images of digits, with the task of classifying between 3s and 8s. The neural network has two convolutional layers and two fully connected layers and uses ReLU activation functions[1]. Bayesian inference is approximated with the Bernoulli dropout method [Gal and Ghahramani, 2016b; Gal and Ghahramani, 2016a], with a dropout probability of 0.2 and 100 Monte Carlo samples at test time, which provides a rough approximation of the model uncertainty for prediction. The locality distribution around image x is defined by randomly zeroing pixels (i.e., setting them to the background value of white color) by first sampling the zeroing probability from a beta distribution and then sampling a binary mask with this probability in independent Bernoulli distributions. The simplified representation has 1 for pixels that are not at the background value and 0 otherwise. 1,000 samples are drawn from the locality distribution and used as data for fitting the explanation models with KL-LIME. The explanation model is a linear logistic regression model with L1 penalty.

The top row of Figure 1 shows an example of explaining an image of 8 that is misclassified by the classifier. The relative explanatory power curve can be used to determine the trade-off between explanation fidelity and complexity. In this case, the curve plateaus around 0.85 explanatory power, showing that it's not possible to attain perfect fidelity with the chosen explanation model. The mean explanation shows the posterior mean of the projected parameters of the explanation model. The pixels in the left hand side of the upper loop of the 8 steer the classification towards an 8 as expected for classifying between 8s and 3s. However, the explanation implies that the model has considered the right hand side parts of the image as pointing the classification to a 3. The variance of the explanation is largest in the left hand side of the upper loop of the digit.

The second row of Figure 1 shows mean explanations at different trade-offs between the fidelity and complexity. The lowest level of explanatory power does not capture the classification particularly well. In the case of images as here, even the most complex explanations (by the measure of how many pixels are active in the linear explanation model) are often readily interpretable, since humans are good at image perception. In many other cases, such as textual data or quantitative covariates, say, in personalized medicine, a proper trade-off between complexity and fidelity would be more important. Finally, the two bottom rows of Figure 1 show individual posterior samples of the explanation model. This allows getting a more complete picture of the model uncertainty reflected in the explanations.

Figure 2 shows another example of an explanation: in this case of an image of a 3 that is misclassified as an 8. The explanation implies that this happens because of the elongated lower left curve in the digit.

5 Conclusion

We presented a method, KL-LIME, to construct local interpretable explanations of Bayesian predictive models by projecting the information in the predictive distribution of the model to a simpler, interpretable probabilistic explanation model. The approach is based on combining ideas from the recent Local Interpretable Model-agnostic Explanations (LIME) method [Ribeiro *et al.*, 2016] and Bayesian projection predictive variable selection. This allows accounting for model uncertainty also in the explanations.

The approach gives a principled way of extending LIME to different types of predictions and explanations as long as we can compute the Kullback–Leibler divergence between the predictive distributions for the original model and the explanation model and minimize it to fit the explanation model. In particular, within this constraint, both the original task and the explanation model can be arbitrarily changed without losing the information theoretical interpretation of the projection for finding the explanation model. The value of the KL divergence can be used as a measure of the fidelity of the explanation.

[1] A slightly customized version of the PyTorch MNIST example is used, `https://github.com/pytorch/examples/tree/master/mnist/`, accessed on May 17th, 2018. This achieves about 99.2% accuracy on the test data.

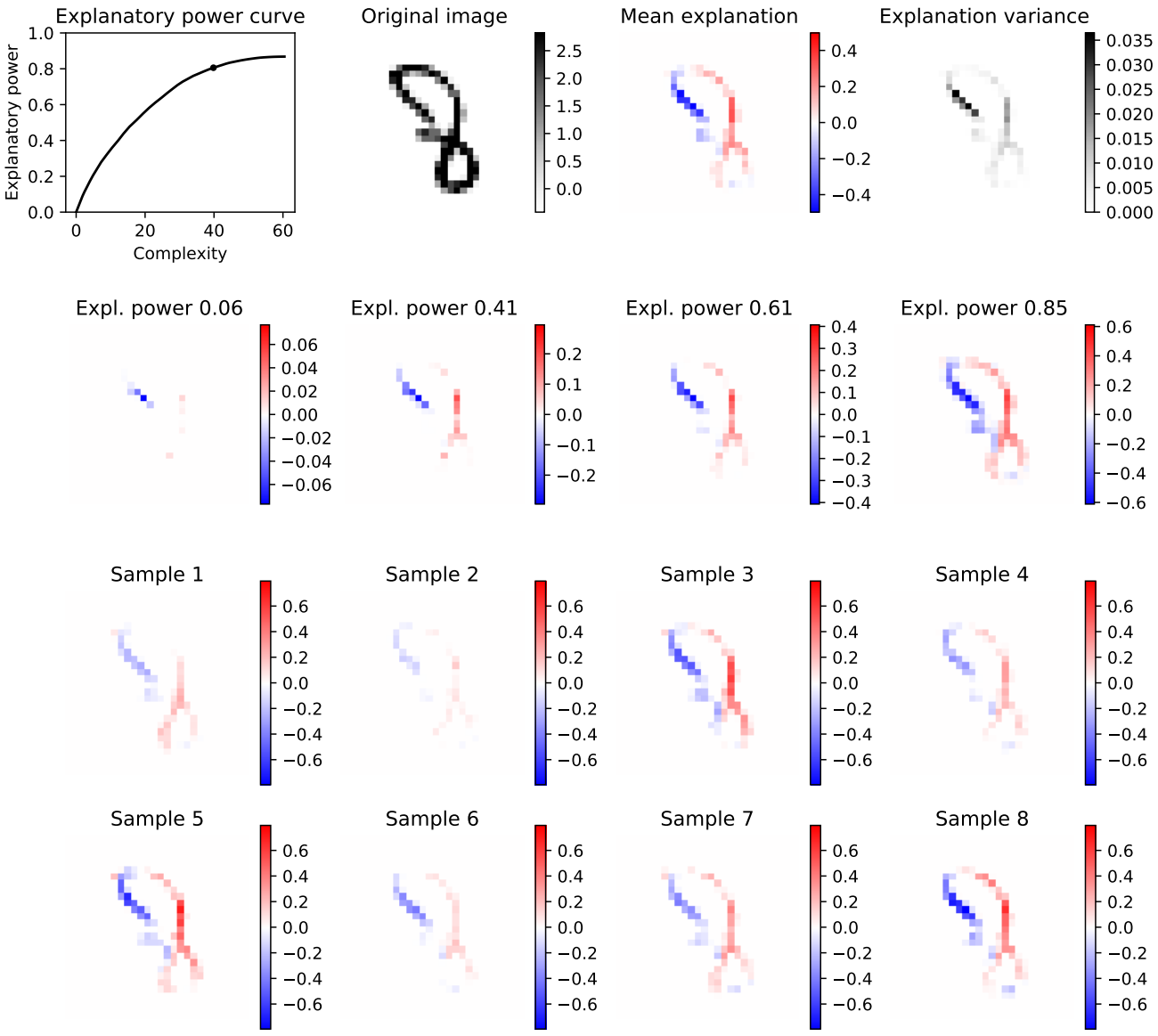


Figure 1: Example explanation of an image of an 8 that was misclassified as 3. Top row: Explanatory power curve shows the relative explanatory power as a function of the explanation complexity (here, the mean number of active pixels in the explanation). Dot shows the complexity chosen for the shown explanation (expl. power of 0.8). Mean explanation gives the posterior mean of the projected explanations and variance gives the uncertainty in the explanation. Second row: Explanations with different complexities and relative explanatory powers. Bottom two rows: Individual posterior samples of explanations.

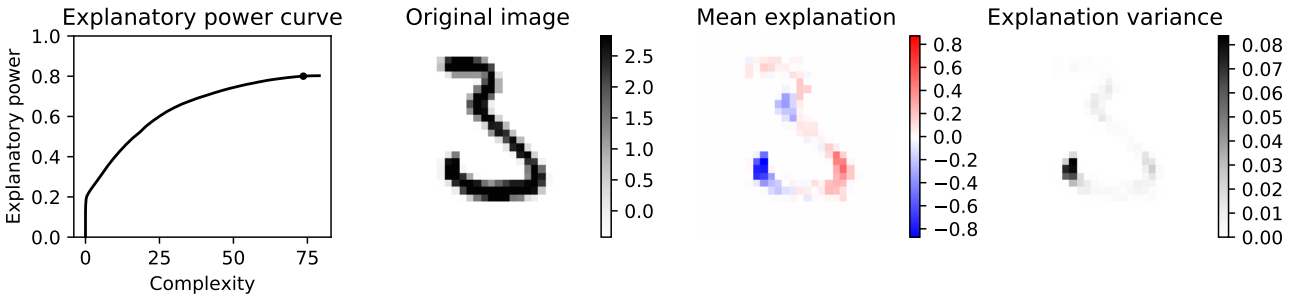


Figure 2: Example explanation of an image of a 3 that was misclassified as 8. See Figure 1 caption for the description of the panels.

References

[Baehrens *et al.*, 2010] David Baehrens, Timon Schroeter, Stefan Harmeling, Motoaki Kawanabe, Katja Hansen, and Klaus-Robert Müller. How to explain individual classification decisions. *Journal of Machine Learning Research*, 11(Jun):1803–1831, 2010.

[Dupuis and Robert, 2003] Jérome A Dupuis and Christian P Robert. Variable selection in qualitative models via an entropic explanatory power. *Journal of Statistical Planning and Inference*, 111(1-2):77–94, 2003.

[Gal and Ghahramani, 2016a] Yarin Gal and Zoubin Ghahramani. Bayesian convolutional neural networks with Bernoulli approximate variational inference. In *4th International Conference on Learning Representations (ICLR) workshop track*, 2016.

[Gal and Ghahramani, 2016b] Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of the 33rd International Conference on Machine Learning*, pages 1050–1059, 2016.

[Goutis and Robert, 1998] Constantinos Goutis and Christian P Robert. Model choice in generalised linear models: A Bayesian approach via Kullback–Leibler projections. *Biometrika*, 85(1):29–37, 1998.

[Lundberg and Lee, 2017] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, pages 4768–4777, 2017.

[Peltola *et al.*, 2014] Tomi Peltola, Aki S Havulinna, Veikko Salomaa, and Aki Vehtari. Hierarchical Bayesian survival analysis and projective covariate selection in cardiovascular event risk prediction. In *Proceedings of the Eleventh UAI Conference on Bayesian Modeling Applications Workshop*, volume 1218, pages 79–88. CEUR Workshop Proceedings, 2014.

[Piironen and Vehtari, 2016] Juho Piironen and Aki Vehtari. Projection predictive model selection for Gaussian processes. In *IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP)*, 2016.

[Piironen and Vehtari, 2017] Juho Piironen and Aki Vehtari. Comparison of Bayesian predictive methods for model selection. *Statistics and Computing*, 27(3):711–735, 2017.

[Ribeiro *et al.*, 2016] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Why should I trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1135–1144. ACM, 2016.

[Tran *et al.*, 2012] Minh-Ngoc Tran, David J Nott, and Chenlei Leng. The predictive lasso. *Statistics and computing*, 22(5):1069–1084, 2012.

Explanatory Masks for Neural Network Interpretability

Lawrence Phillips[1], Garrett Goh[2], Nathan Hodas[1],

[1] Computing & Analytics Division, Pacific Northwest National Lab
[2] Advanced Computing, Mathematics & Data Division, Pacific Northwest National Lab
{lawrence.phillips, garret.goh, nathan.hodas}@pnnl.gov

Abstract

Neural network interpretability is a vital component for applications across a wide variety of domains. In such cases it is often useful to analyze a network which has already been trained for its specific purpose. In this work, we develop a method to produce explanation masks for pre-trained networks. The mask localizes the most important aspects of each input for prediction of the original network. Masks are created by a secondary network whose goal is to create as small an explanation as possible while still preserving the predictive accuracy of the original network. We demonstrate the applicability of our method for image classification with CNNs, sentiment analysis with RNNs, and chemical property prediction with mixed CNN/RNN architectures.

1 Introduction

Network interpretability remains a required feature for machine learning systems in many domains. A great deal of recent work has been done to try to shed light on deep learning models, producing methods that create local explanations [Ribeiro *et al.*, 2016], following gradients [Selvaraju *et al.*, 2016], investigating input perturbations [Fong and Vedaldi, 2017], and even generate textual explanations [Hendricks *et al.*, 2016]. Many of these techniques involve creating an input *mask*, assigning weights to individual inputs such that a weighted visualization of the input can be created.

In this work, we explore an alternate method for generating such an explanatory mask. Our goal is to generate an explanatory input mask with two properties: 1) the explanation should be as minimal as possible, 2) the explanation should highlight the portions of the input most necessary to ensure predictive accuracy. To keep the explanation mask as small as possible we make use of a variety of regularization techniques. To keep predictive accuracy we force the mask to generate accurate outputs when passed through the original network.

2 Explanation Masks

Given a pre-trained network, the goal of the explanation mask is to identify which inputs are most necessary for creating accurate decisions. To do this, we make use of a secondary, *explanation* network, which learns to generate explanation masks in an end-to-end fashion. By training the network, we are able to ensure that the generated mask does indeed carry predictive accuracy. Further, by implementing regularization over the mask, we ensure that the mask is as minimal as possible while still being accurate. This trade-off between accuracy and the size of the mask is one which can be explored on a per-case basis by changing the amount of regularization imposed.

The structure of the explanation network can be of any form fit to the problem task which leaves the method quite flexible. A schematic of the overall system is given in Figure 1. First, the pre-trained network is split into a feature extraction and classification component. This can be done arbitrarily, but can often be done after a feature *bottleneck* in the original network. The output of this feature extraction step is then fed into the explanation network. Given the features describing the input, the explanation network can then generate a mask of the same size as the input. The mask should have weights between 0 and 1 and is multiplied element-wise against the original input.

Once the input has been masked, it can then be fed through the entirety of the original network, both feature extraction and classification components, to produce a final output. To train the explanation network, we freeze the weights of the original network. This ensures that the explanation mask generated is an accurate reflection of what inputs are necessary for the pre-trained network. Training can be accomplished end-to-end through the use of backpropagation over the non-frozen weights.

2.1 Related Work

Over the past few years there have been many attempts to shed light on the inner workings of neural networks. One of the most general approaches has been the introduction of attention mechanisms as initially popularized in machine translation [Bahdanau *et al.*, 2015]. Attention mechanisms operate similarly to our explanation masks in that they result in a mask where every element in the input is weighted between 0 and 1 and combined through a weighted sum or product. Traditional attention is trained as a single network which benefits from the end-to-end nature of deep learning but also requires attention to be added initially. This means that while atten-

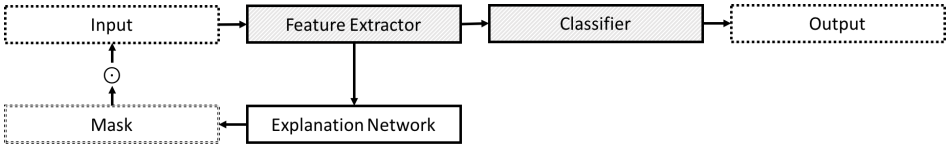


Figure 1: Explanation mask architecture. The pre-trained network on top is divided into a feature extraction and classification component. The output of the feature extractor feeds into an explanation network, producing a mask that is multiplied element-wise with the input. The masked input then is fed through the original network to produce a final output. Shading indicates frozen weights.

tion mechanisms are often worth incorporating when creating a model, they cannot be used to explain a pre-trained model's decisions if the original model did not have an attention mechanism.

An intriguing alternative to basic attention is the ability to generate natural language explanations to account for a model's decisions. This idea is explored in Hendricks et al. [2016], where an image classifier was trained along with a sentence generator in order to create a short, explanatory sentence along with every classification. A limitation of this work is the weakly-enforced correspondence between image and explanation, which can result in generic explanations that do not reference portions of the image. This is refined in Hendricks et al. [2017] where explanations are more carefully trained to be tailored to a specific image. While powerful, these models require complicated training through reinforcement learning and are only applicable for classification datasets which contain ground truth explanations for each class.

A last group of methods attempt to explain classifier predictions through perturbing data and modeling how this affects the base model. This is explored in Fong et al. [2017] where images are perturbed through effects such as blurring in a systematic fashion to reduce the model's classification accuracy. In this way, the authors are able to identify regions of the image which are important to classification. A similar method was introduced in Selvaraju et al. [2016] by exploring model gradients. Lastly, LIME [Ribeiro *et al.*, 2016] has been a popular approach for creating a "interpretable" models, such as sparse linear models, that are locally consistent with more complex black-box models. The downside of this method is that it necessarily removes information from the original model and some have argued that simple models are less interpretable than one might original assume [Lipton, 2016].

3 Experiments

3.1 Image Classification

We make use of the CIFAR10 image classification dataset [Krizhevsky and Hinton, 2009] to test our models ability on visual data. CIFAR10 is made up of 60,000 images from 10 classes, with 50,000 used for training and 10,000 as a test set. As a base model, we use ResNet164 v2 [He *et al.*, 2016b] which is a 164-layer residual network. After training, the model achieves a test set accuracy of 94.04%.

The ResNet architecture generates a vector of length 256 to represent the input image before making a final dense softmax decision regarding its category. We take this bottleneck vector as the input to the explanation network. The 256-dimension vector is reshaped into a 16×16 matrix and passed through two convolutional blocks. Each block is made up of 4 2D convolutional blocks with *tanh* activations. Padding ensures the size of the image remains constant and a filter size of 64 is used. A residual connection [He *et al.*, 2016a] is used where the first and final convolutional layers have their activations summed. After the first block, the mask is upsampled to 32, the size of the original image. After the second block, the explanation network ends with a single 2D convolution with filter size 1 and sigmoid activation.

As described above, the mask is multiplied by the original input and is applied equally across all 3 color channels. Sparsity in the mask is encouraged through regularization with l2 = 1e-4. We present original CIFAR10 images along with their learned masks in Figure 2

3.2 Sentiment Analysis

To assess how well the explanation masks work on natural language tasks, we look at the IMDB review sentiment task [Maas *et al.*, 2011]. This involves 50,000 movie reviews rated as either positive or negative. We preprocess the input so that it has a vocabulary size of 20,000 and all infrequent words are replaced with a special OOV token. We further reduce the cap the maximum length of each review at 500 words. Our base model uses 100-dimensional, pretrained GloVe embeddings with 40% word-level dropout to prevent overfitting. This is followed by a bidirectional GRU layer of width 256. The output of this layer is averaged over timesteps and passed to a 128-dimension fully connected layer with relu activation and a final dense softmax performs the final classification. We train the network to optimize binary crossentropy with the Adam optimizer using the standard 50/50 train/test split. After training for 10 epochs we achieve a test set accuracy of 84.7%.

For our explanation network we use the full output of the bidirectional GRU layer as input. The explanation mask is calculated by using a 2-layer bidirectional GRU of width 100. Each timestep's output is passed to a fully connected layer of size 256 with relu activation before a linear layer outputs a single value for the timestep, representing the weight of that word in the mask. The mask is regularized by penalizing its entropy. After training for 10 epochs we achieve a test set accuracy of 81.8%. In Figure 3 we present example sentences with the size of each word scaled to the mask weights. We also explore how well the model is able to capture general positive and negative trends by looking at which words receive the most attention in positive and negative reviews, which we display on the right of Figure 3.

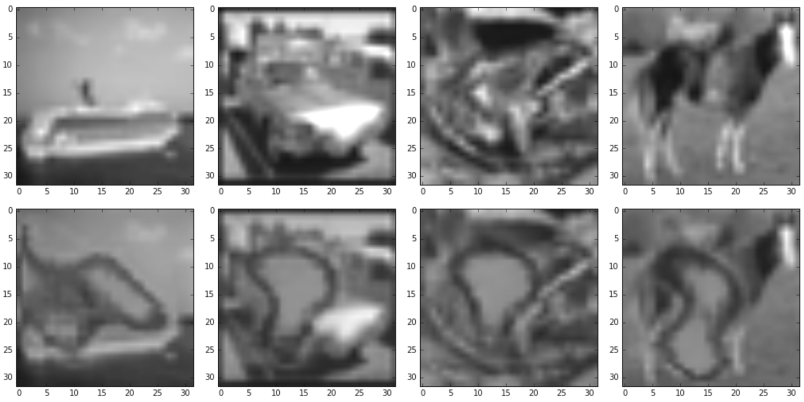
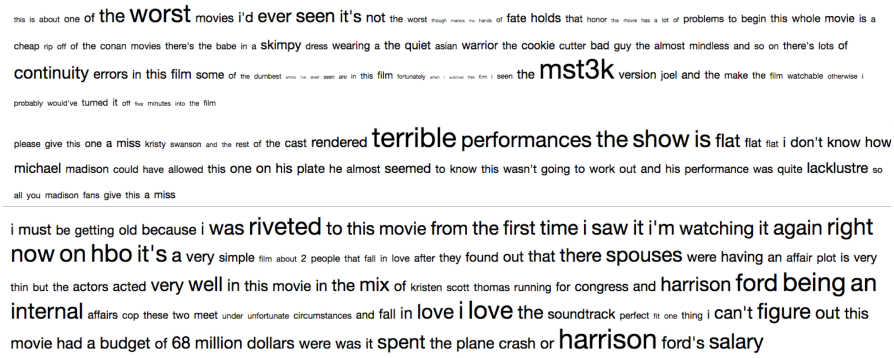


Figure 2: CIFAR10 images (top) along with their learned explanation masks (bottom).



this is about one of the worst movies i'd ever seen it's not the worst though makes the hands of fate holds that honor this movie has a lot of problems to begin this whole movie is a cheap rip off of the conan movies there's the babe in a skimpy dress wearing a the quiet asian warrior the cookie cutter bad guy the almost mindless and so on there's lots of continuity errors in this film some of the dumbest errors i've ever seen are in this film fortunately when i watched this film i seen the mst3k version joel and the make the film watchable otherwise i probably would've turned it off five minutes into the film

please give this one a miss kristy swanson and the rest of the cast rendered terrible performances the show is flat flat flat i don't know how michael madison could have allowed this one on his plate he almost seemed to know this wasn't going to work out and his performance was quite lacklustre so all you madison fans give this a miss

i must be getting old because i was riveted to this movie from the first time i saw it i'm watching it again right now on hbo it's a very simple film about 2 people that fall in love after they found out that there spouses were having an affair plot is very thin but the actors acted very well in this movie in the mix of kristen scott thomas running for congress and harrison ford being an internal affairs cop these two meet under unfortunate circumstances and fall in love i love the soundtrack perfect fit one thing i can't figure out this movie had a budget of 68 million dollars were was it spent the plane crash or harrison ford's salary

Positive	Negative
best	boring
great	waste
very	worst
though	awful
well	terrible

Figure 3: Left: IMDB reviews with words scaled in height based on explanation mask weights. Right: Top 5 words of positive and negative sentiment.

From the example sentences in Figure 3, one can see that the model is able to attend to words and phrases which indicate both positive and negative sentiment. In the first sentence, the model recognizes that the word *mst3k*, referencing the TV show "Mystery Science Theater 3000", is highly indicative of negative sentiment. In the final example, the name *Harrison* is identified as highly positive. Attention is more often placed over phrases, indicating that the model has identified that combinations of words are indicative of sentiment. Again in the final example, we see the phrase "right now on HBO" given high attention. In the second example, attention spans two sentences over "rendered terrible performances. The show is flat".

3.3 Chemical Solubility

As a final proof-of-concept, we explore the ability of our method to work for networks trained on small data. To do this we make use of the SMILES2vec model [Goh *et al.*, 2017] to predict chemical solubility on the ESOL dataset [Wu *et al.*, 2018]. The ESOL dataset is quite small, with only 1,128 data points and differs from the previous tasks in that it involves regression rather than classification.

The SMILES2vec model is a mixed CNN-GRU architecture which takes in chemical compounds as strings represented using SMILES [Weininger, 1988] and predicts chemical solubility. The network is structured as in Figure 4a. The network is trained as in [Goh *et al.*, 2017] and achieves a test RMSE of 0.846. The explanation network takes as input the embedded SMILES representation and then outputs a character-level attention mask which is applied equally across all embedding dimensions for each character. The network is structured as a 20 layer convolutional residual network with SELU non-linearities. Padding is used for each convolution to ensure the input size remains constant. The final layer of the explanation network is a 1D convolution of length 1 with batch normalization followed by a softplus activation. We regularize the explanation network's output through L1 and L2 loss set to 1e-3 and 1e-4 respectively. The explanation network is trained until convergence and has a test RMSE of 0.831, slightly better than the original network alone.

To test the model's interpretations, we examined the output qualitative and quantitatively. In Figure 4b, we highlight the explaination mask, with darker circles indicating more attention from the network. For soluble molecules, the network properly attends to atoms which are part of hydrophilic groups, which increase solubility. The opposite is true of insoluble molecules, as in the bottom right, where the network attends to the Cl molecule as part of a hydrophobic group. To quantify these patterns, we split the data between soluble ($\leq$-1.0) and insoluble ($\geq$-5.0) compounds. The network should

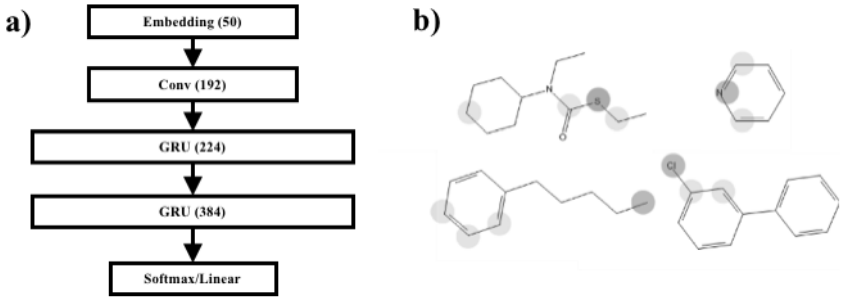


Figure 4: (a) Illustration of the SMILES2vec CNN-GRU architecture. (b) The explanation mask of SMILES2vec validates established knowledge by focusing on atoms of known hydrophobic and hydrophilic groups, colored circles of increasing darkness indicate increasing attention.

attend more to the atoms O and N for soluble compounds and C, F, Cl, Br, and I for insoluble. We identified the top-3 characters attended to in each explanation mask and computed the top-3 accuracy for identifying these particular atoms. The network achieves a top-3 accuracy of 88%, indicating that the attention mask does reliably identify atoms known to affect solubility.

4 Conclusion

Through experiments over three diverse domains and network architectures we have shown how pre-trained networks might be analyzed by means of explanation masks. Although generating explanation masks requires a small amount of training for the explanation network, the method is quite flexible in its structure and can be applied to most pre-trained networks. While there are many possible approaches to model explanation, ours explicitly focuses on the aspects of the input which are most necessary for accurate prediction.

References

[Bahdanau *et al.*, 2015] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *ICLR*, 2015.

[Fong and Vedaldi, 2017] Ruth C Fong and Andrea Vedaldi. Interpretable explanations of black boxes by meaningful perturbation. *arXiv preprint arXiv:1704.03296*, 2017.

[Goh *et al.*, 2017] Garrett B Goh, Nathan O Hodas, Charles Siegel, and Abhinav Vishnu. Smiles2vec: An interpretable general-purpose deep neural network for predicting chemical properties. *arXiv preprint arXiv:1712.02034*, 2017.

[He *et al.*, 2016a] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[He *et al.*, 2016b] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European Conference on Computer Vision*, pages 630–645. Springer, 2016.

[Hendricks *et al.*, 2016] Lisa Anne Hendricks, Zeynep Akata, Marcus Rohrbach, Jeff Donahue, Bernt Schiele, and Trevor Darrell. Generating visual explanations. In *European Conference on Computer Vision*, pages 3–19. Springer, 2016.

[Hendricks *et al.*, 2017] Lisa Anne Hendricks, Ronghang Hu, Trevor Darrell, and Zeynep Akata. Grounding visual explanations. In *NIPS Interpretable ML Symposium*, 2017.

[Krizhevsky and Hinton, 2009] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

[Lipton, 2016] Zachary C Lipton. The mythos of model interpretability. *arXiv preprint arXiv:1606.03490*, 2016.

[Maas *et al.*, 2011] Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, volume 1, pages 142–150, 2011.

[Ribeiro *et al.*, 2016] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1135–1144. ACM, 2016.

[Selvaraju *et al.*, 2016] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. *See https://arxiv. org/abs/1610.02391 v3*, 7(8), 2016.

[Weininger, 1988] David Weininger. Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules. *Journal of chemical information and computer sciences*, 28(1):31–36, 1988.

[Wu *et al.*, 2018] Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing, and Vijay Pande. Moleculenet: a benchmark for molecular machine learning. *Chemical Science*, 9(2):513–530, 2018.

Transparency Communication for Reinforcement Learning in Human-Robot Interaction

David V. Pynadath[1], **Ning Wang**[1], **Michael J. Barnes**[2],
[1] Institute for Creative Technologies, University of Southern California
[2] US Army Research Laboratory
pynadath@usc.edu, nwang@ict.usc.edu, michael.j.barnes.civ@mail.mil

Abstract

Autonomous systems need to learn from their experience to improve their decisions. Machine-learning methods, particularly reinforcement learning, have successfully applied quantitative probabilities and utilities to a variety of real-world domains, including allowing robots to improve from their mistakes. The complexity of the domains that robots operate in and the inherent complexity of their decisions make understanding the inner workings of robotics systems increasingly challenging for their human teammates. While transparency communications have shown to alleviate such problems, the additional capability of self-improving, enabled by reinforcement learning, is likely to complicate the robot's effort to reason transparently with human teammates. In this paper, we discuss the design of both model-based and model-free reinforcement learning for the robots in a human-robot simulation testbed and the design of transparency communications that unpacks the components of the robot's decision-making and learning process.

1 Introduction

Autonomous systems operating in complex domains must reason under uncertainty, prioritize conflicting goals, and adapt to unpredictable environments. While autonomous systems have increased their capabilities over the decades, their potential is not often realized when teamed up with humans [Parasuraman and Riley, 1997]. Due to the complexity of the domains that autonomous systems now operate in and the inherent complexity of the decisions they make, understanding the inner workings of such systems becomes increasingly challenging for the humans. Such "black box" phenomena often result in disuse or over-reliance of the autonomous systems [Parasuraman and Riley, 1997]. In human-human teams, communication targeting the gaps in understanding can help improve shared situation awareness, foster trust relationships, and enhance team performance. Inspired by the research in human-human teams, researchers have designed both hand-crafted and automatically generated explanations to achieve similar outcomes in human-automation teams [Dzindolet *et al.*, 2003; Wang *et al.*, 2016].

However, the autonomous systems used in transparency communication research often lack the ability to learn from their experience or from data to improve their decisions. The very definition of autonomy, in the field of artificial intelligence, means that a system should function beyond the initial knowledge encoded by the designers and must be able to adapt in the face of the unforeseen and the unknown [Russell and Norvig, 2016]. Machine-learning methods have been widely applied to the design of automation, including virtual agents and robots. Reinforcement Learning (RL), in particular, is a powerful machine-learning method that has successfully applied quantitative probabilities and utilities to a variety of real-world domains [Kaelbling *et al.*, 1996]. RL is based on the idea that an agent can autonomously discover an optimal behavior through trial-and-error interactions with its environment. The agent explores the space of possible strategies and receives feedback on the outcomes of the choices made. From this information, a "good"—or ideally optimal—policy (i.e., strategy or controller) can be deduced [Sutton and Barto, 1998; Kober *et al.*, 2013]. RL's algorithms for computing long-term expected values can provide autonomous agents with optimal sequential decision policies.

While RL is based on the simple idea of action-reward mappings, elements of RL can be complex due to their quantitative values and the iterative process by which they are computed. The rich representation and complex reasoning from RL, which provide useful performance guarantees for software agents, also present a significant obstacle to human understanding of, for example, how the value functions are constructed, how the algorithms update the value function, and how such updates impact the action/policy chosen by the agent. Without such understanding, a human operator is likely to fall into the same pitfall of misuse or disuse of the agents [Parasuraman and Riley, 1997]. Therefore, for autonomous learning agents to play an effective role in human-machine teams, they must make their RL-generated decisions understood by their human operators and teammates.

Automatically generated explanations have provided such understanding in other areas of artificial intelligence (AI) [Swartout *et al.*, 1991], suggesting the potential for similar understanding of RL-based AI. Model-based RL first learns a quantitative model in the form of Partially Observ-

able Markov Decision Processes (POMDPs), which contain probabilistic action and sensor models, utility-based goal priorities, etc. [Kaelbling *et al.*, 1998] that could facilitate explanations. For real-world domains, the size and complexity of quantitative models like POMDPs are more likely to overwhelm human operators, rather than inform them. In our previous work, we have developed algorithms to automatically generate explanations for decisions made by POMDP-based agents. These algorithms were among the first research efforts to apply the methodology of explainable AI (XAI) to make quantitative agent-based decision models explainable and transparent to human users [Wang *et al.*, 2016].

In this paper, we discuss the design of RL for the robots in our human-robot reconnaissance task. We then discuss the design of explanations that can potentially make RL components transparent to human teammates.

2 Related Work

As AI systems become increasingly complex and bestowed with the capability to learn and evolve, their decision-making mechanisms become increasingly opaque to their human users. With the maturation of many AI-powered technologies, they are transitioning from operating from behind-the-scenes, to directly interfacing with humans. The "black box" issue of many AI systems has hindered the growth of the capability of human-AI systems teams. This problem has garnered much attention, and much research effort has been exerted to create an interface to make the decision process of AI algorithms more transparent. For example, in [Hendricks *et al.*, 2016], researchers used a combination of convolutional neural networks (CNN) and recurrent neural networks (RNN) to recognize objects in images and generate image captions based on recognized objects. Other efforts have focused not on unpacking the "black box" itself, but on generating explanations of decisions (e.g., classifications) from the "black box" based on the input (e.g., instances from dataset) and the output [Ribeiro *et al.*, 2016].

Our current work follows a long history of automated XAI mechanisms, especially within the context of expert systems [Swartout and Moore, 1993]. While most of this work operated on rule- and logic-based systems, there has been more recent work on generating explanations based on Markov Decision Problems (MDPs) [Elizalde *et al.*, 2008] and Partially Observable MDPs (POMDPs) [Wang *et al.*, 2016]. The existing evidence is encouraging as to the potential success of applying a general-purpose explanation on top of an agent's decision-making process. In addition to explaining POMDP-based decisions in simulated robots, there has been recent work on generating explanations and justifications for decisions by simulated and physical robots using decision trees [Sheh, 2017], answering human queries of the robot's policies [Hayes and Shah, 2017], and making the robot's planning process easier for humans to understand and predict [Zhang *et al.*, 2017].

There is a rapidly growing body of work on applying reinforcement learning to enable robots to improve from their mistakes, e.g., [Matarić, 1997; Smart and Kaelbling, 2002]. While such learning is likely to complicate the robot's effort to reason transparently with human teammates, it does provide an opportunity to repair trust that has been damaged by robot errors. Our investigation into the interaction between explanations and trust repair is inspired by work on methods for the latter within organizations [Lewicki, 2006; Schweitzer *et al.*, 2006]. Prior research has similarly examined the effectiveness of these trust-repair strategies within HRI [Robinette *et al.*, 2015]. This work found that the timing and combination of trust-repair actions was critical to effectively maintaining trust.

As RL has moved into more and more domains, there have also been more and more investigations into XAI with RL systems. For example, in [Iyer *et al.*, 2018], researchers produced visualizations of the state and behavior from agents using deep reinforcement learning networks. Within a specifically human-robot domain, researchers proposed Instruction-based Behavior Explanation (IBE) to allow human users to interactively provide input into the RL process while receiving explanations of the estimated outcome of such input. [Fukuchi *et al.*, 2017]. In our previous work, we have experimented with robot transparency communications that acknowledges errors and promises to improve [Wang *et al.*, 2018]. Such communication did not influence human trust and team performance, possibly due to a lack of explanation of *how* the robot plans to improve its decisions. In the work presented here, we focus on applying model-based and model-free reinforcement learning to update (and hopefully improve) the robot's decision-making and on designing explanations that communicate such an updating process to the human in an effort to build transparency and repair trust.

3 Human Robot Interaction Testbed

We conduct our investigation of explainable AI within an online HRI testbed that we have used to gather human behavior data when interacting with a simulated robot [Wang *et al.*, 2015].

3.1 POMDP Model of Testbed

The robot in this testbed bases its decisions on a POMDP [Kaelbling *et al.*, 1998] model, which is a tuple, $\langle S, A, P, \Omega, O, R \rangle$, that we describe here in terms of the HRI testbed scenario [Wang *et al.*, 2015]. In it, a human teammate works with a robot in reconnaissance missions to gather intelligence in a foreign town. Each mission involves the human teammate searching buildings in the town. The robot serves as a scout, scans the buildings for potential danger, and relays its observations to the teammate. Prior to entering a building, the human teammate can choose between entering with or without equipping protective gear. If there is danger present inside the building, the human teammate will be injured if not wearing the protective gear, causing the team to incur a high time penalty. However, it also takes time to put on and take off protective gear (although much less time than the injury penalty). So the human teammate is incentivized to consider the robot's observations before deciding how to enter the building. The simulated robot has an NBC (nuclear, biological and chemical) weapon sensor, a camera that can detect hostile gunmen, and a microphone that can listen to discussions in foreign language.

The state, S, consists of objective facts about the world, some of which may be hidden from the agent itself. We use a *factored representation* [Boutilier *et al.*, 2000] that decomposes these facts into individual feature-value pairs, such as the robot's current location, as well as the presence of dangerous people or chemicals in the buildings to be searched. The state may also include feature-value pairs that represent the health level of its human teammate, any current commands, and the accumulated time cost so far.

The available actions, A, correspond to the possible decisions the agent can make. Given the proposed mission, the agent's first decision is where to move to next. Upon completing a search of a building, an agent can make a decision as to whether to declare a location as safe or unsafe for its human teammate. For example, if a robot believes that armed gunmen are at its current location, then it will want its teammate to take adequate preparations (e.g., put on body armor) before entering. Because there is a time cost to such preparations, the robot may instead decide to declare the location safe, so that its teammate can more quickly complete their own reconnaissance tasks.

The transition probability function, P, represents the effects of the agent's actions on the subsequent state. In the current testbed, the robot's movement actions always succeed. Recommendation actions, on the other hand, can affect the health and happiness of its human teammate, although only stochastically, as a person may not follow the recommendation.

The "partial observability" of a POMDP is specified through a set of possible observations, Ω, that are probabilistically dependent (through the observation function, O) on the true state of the world. Different observations may have different levels of noise. For example, an agent may be able to use GPS to get very accurate readings of its own location. However, it may not be able to detect the presence of hostile gunmen or dangerous chemicals with perfect reliability or omniscience. Instead, the agent will receive local readings about the presence (or absence) of threats in the immediate vicinity. For example, if dangerous chemicals are present, then the robot's chemical sensor may detect them with a high probability. There is also a lower, but nonzero, probability that the sensor will not detect them. In addition to such a false negative, we can also model a potential false positive reading, where there is a low, but nonzero, probability that it will detect chemicals even if there are none present. By controlling the observations that the agents receive, we can manipulate their ability level in our testbed.

Partial observability gives the robot only a subjective view of the world, where it forms beliefs about what it thinks is the state of the world, computed via standard POMDP state estimation algorithms. For example, the robot's beliefs may include its subjective view on the presence of threats, in the form of a likelihood (e.g., a 33% chance that there are toxic chemicals in the farm supply store). Again, the robot would derive these beliefs from prior beliefs about the presence of such threats, updated by its more recent local sensor readings. Due to the uncertainty in its prior knowledge and sensor readings (not to mention its learning), the robot's beliefs are likely to diverge from the true state of the world. By decreasing the accuracy of the robot's observation function, O, we can decrease the accuracy of its beliefs, whether receiving correct or incorrect observations. In other words, we can also manipulate the robot's ability by allowing it to over- or under-estimate its sensors' accuracy.

The human-machine team's mission objectives are captured by the reward function, R, which maps the state of the world into a real-valued evaluation of benefit for the agents. In our example domain, the robot receives the highest reward when the surveillance is complete. It will also receive higher reward values when its teammate is alive and unharmed. This reward component punishes the agents if they fail to warn their teammates of dangerous buildings. Finally, the agent will receive a slight negative reward for the amount of time that passes. This motivates the agents to complete the mission as quickly as possible.

By constructing such a POMDP model of the mission, the agent can autonomously generate its behavior by determining the optimal action based on its current beliefs, b, about the state of the world [Kaelbling *et al.*, 1998]. The agent uses a (bounded) lookahead procedure that seeks to maximize expected reward by simulating the dynamics of the world from its current belief state across its possible action choices. It will combine these outcome likelihoods with its reward function and choose the option that has the highest expected reward.

3.2 Explanation Generation for POMDPs

The elements $\langle S, A, P, \Omega, O, R \rangle$ of a POMDP model correspond to concepts that people are likely to be familiar with. By exposing different components of an agent's model, we can make different aspects of its decision-making transparent to human teammates. In prior work, we created static templates to translate the contents of a POMDP model into human-readable sentences [Wang *et al.*, 2016]. We create such templates around natural-language descriptions of each state feature and action. We then instantiate the templates at runtime with prespecified functions of the agent's current beliefs (e.g., probability of a state feature having a certain value). The following list illustrates the templates we created for each POMDP component, using specific runtime instantiations to show the final natural-language text provided to a human participant:

- ***S*:** The agent can communicate its current beliefs about the state of the world, e.g., "I believe that here are no threats in the market square." The agent could also use a standard POMDP probabilistic belief state to communicate its uncertainty in that belief, e.g., "I am 67% confident that the market square is safe."
- ***A*:** An agent can make a decision about what route to take through its search area, e.g., "I am proceeding through the back alley to the market square."
- ***P*:** An agent can also reveal the relative likelihood of possible outcomes based on its transition probability model, e.g., "There is a 33% probability that you will be injured if you follow this route without taking the proper precautions."

Ω: Communicating its observation can reveal information about an agent's sensing abilities, e.g., "My NBC sensors have detected traces of dangerous chemicals."

O: Beyond the specific observation it received, an agent can also reveal information about its observation model, e.g., "My image processing will fail to detect armed gunmen 30% of the time."

R: By communicating the expected reward outcome of its chosen action, an agent can reveal its alignment (or lack thereof) with the mission objective, contained in its reward function, e.g., "I think it will be dangerous for you to enter the informant's house without putting on protective gear. The protective gear will slow you down a little." The template here relies on factored rewards, allowing the agent to compute separate expected rewards, $E[R]$, over the goals of keeping its teammate unharmed and achieving the mission as quickly as possible.

3.3 Results from POMDP-Based Explanation

We have used this testbed to gather human behavior data when interacting with the robot under combinations of these POMDP-based explanation algorithms. We designed explanations that selected different aspects of the robot's decision-making process at different level of details. Evaluations of such explanations have indicated that when the robot provides an explanation to justify its decisions (e.g., by providing a numerical confidence level of its decisions or a summary of findings from its sensors), the human-robot team performed better on simulated missions, compared to when no explanations were given [Wang *et al.*, 2016].

More relevant to a learning robot, we augmented explanations with a trust-repair strategy inspired by prior work in organizational trust: an acknowledgement of a mistake, paired with a promise to improve before the next mission [Schweitzer *et al.*, 2006]. This acknowledgment made the human teammate trust the robot more when the robot did not provide any explanation of its previous (and mistaken) decision. However, overall, such communication did not make any impact on the team performance, across a variety of explanation content [Wang *et al.*, 2018]. This lack of impact was most likely due to the fact that participants interacted with each specific robot for only one mission, so they would never be able to observe any improvement (so we never implemented any either). The natural next step is to enrich our agent to perform this self-improvement during the mission itself. This result highlighted the importance of unpacking the machine learning process of the robot and making it transparent to a robot's human teammate.

4 Reinforcement Learning in the HRI Testbed

4.1 Model-Based RL

Given the original POMDP model, it is a relatively easy step for the agent to update that model using reinforcement learning. As in most RL domains, we assume that S, A, and Ω are known a priori. That leaves P, O, and R as the functions to be learned. In this domain, R is a pre-defined mission objective, so no learning occurs there.

The robot's movement is deterministic, so the only part of the transition probability, P, to be learned is the probability that the teammate will follow the robot's recommendation. In the decision cycle after giving its recommendation, the robot observes whether its human teammate followed it or not. It can then use this signal to modify the probability table within P corresponding to its recommendation action and the subsequent effect on the teammate's life. For example, if the teammate ignored the robot's recommendation to wear protective gear and was injured as a result, then the robot could decrease $P(\neg$*human injured*, *recommend protective gear*, $\neg$*human injured*$)$ and increase $P(\neg$*human injured*, *recommend protective gear*, *human injured*$)$. We do not experiment with such learning, as such a change would make the robot less likely to recommend protective gear in the future, whereas it should instead attempt an alternative explanation to convince the teammate to follow its recommendation.

We instead focus our model-based RL on the observation function, O. In many of our prior experiments, we have given the robot a broken camera that fails to detect gunmen in a building, even when present. Allowing such a robot to update O based on experience would help it overcome such a lack of reliability. When the human teammate enters a building, the true state of any threat within is revealed. At this point, the robot will know which of its sensor readings were correct and which not. For example, consider a false negative where the robot's camera (and image-processing system) fail to detect any hostile occupants, but where its microphone (and natural-language processing system) does detect them. The robot can increase $O($*gunmen*, *camera* $=$ *no gunmen*$)$ and $O($*gunmen*, *microphone* $=$ *gunmen*$)$. As a result, the robot will be more likely to recommend protective gear in future buildings when its microphone is positive for gunmen and its camera is negative.

Model-based RL can update the parameters of the robot's POMDP model in this way. It can then use this now-dynamic POMDP model to feed the same explanation generation used for the static POMDP models in our past work. In particular, if the agent is updating its O function, then an O template (as described in Section 3.2) could make the revision transparent by saying "My image processing will fail to detect armed gunmen 45% of the time.". The teammate will thus be informed as to the changes in O resulting from the model-based RL.

4.2 Model-Free RL

Alternatively, the agent can use the POMDP model to compute an initial value function, but then use model-free RL to update those values. In other words, the robot maintains only a set of Q values, $Q(b, a)$, throwing away the POMDP model after initializing them. In our particular domain, the agent's belief state, b, is determined by its sensor readings in the current building. We can thus represent b as a tuple $\langle c, n, m\rangle$ of the robot's camera, NBC sensor, and microphone readings, respectively.

To initialize $Q(\langle c, n, m\rangle, a)$, we first solve the robot's prior POMDP model to arrive at a value function, V, over belief states, b, and actions, a. We can then iterate through each

combination of sensor readings, $\langle c, n, m \rangle$, compute the belief state derived from those sensor readings, b, and then assign the corresponding value to our Q values: $Q(\langle c, n, m \rangle, a) \leftarrow V(b, a)$.

We can then follow a standard model-free RL formulation to update these Q values based on the robot's experiences. The robot will receive a reward signal after its recommendation, based on whether the human teammate suffered any injury and whether time was lost from putting on protective gear. This reward signal will be used to update the Q values for the current sensor readings. Over time, these Q values should approach the optimal values for recommending protective gear based on the sensor readings of threats in the building.

It is important to note that these Q values will account for the effect of broken sensors, disobedient teammates, etc., but with no explicit representation of these factors. For example, if the robot's camera is broken so that it never detects gunmen, the Q values will converge to a table whose values are change based on only the NBC sensor on microphone. However, there will be no explicit representation of the observation function, O, as there was for the model-free case.

5 Explanation Generation for RL

Enabling the agent to learn should enhance its domain-level performance and potentially improve team performance. However, the unpredictability introduced by allowing the agent to change its model (and its decision-making) can lead to distrust from human teammates. It therefore becomes important for the agent to be able to make its learning transparent, as well as its decision-making.

5.1 POMDP-Based Explanation

We first seek to leverage our existing POMDP-based explanation templates. When our robot is using model-based RL, it is trivial to apply the templates from Section 3.2. At each stage, the robot has a POMDP model of the world, and it uses that POMDP to make its decisions. Therefore, any learned components (such as the O example, given in Section 4.1) can be directly fed into the POMDP templates to reveal the agent's current model. Of course, the RL agent's POMDP model will change over time, causing these explanations to change over time as well. However, the dynamics of these explanations can potentially reveal enough information for the teammates' to understand why the robot behavior has changed.

On the other hand, model-free RL (as described in Section 4.2) does not maintain a POMDP model, but rather just the Q values that specify the values associated with all possible actions in a given belief state. Therefore, we cannot directly apply our POMDP explanation templates. However, unless our POMDP modeling structure is erroneous, then there are likely to be many POMDPs that are consistent with the policy arrived at by model-free RL. Therefore, if we can find a POMDP that matches the learned policy, then we can potentially use it as the input to our POMDP explanation templates, just as we can for model-based RL.

To find such matching POMDPs, we first consider a set of candidate matches. One such set is the set of possible POMDPs that can be learned by our model-based RL alternative (as described in Section 4.1. In our HRI domain, model-based RL can vary only O, so we need to consider only the variations of our initial POMDP that differ in O. We can discretize the space of possible O functions to arrive at a finite set of candidate POMDPs.

For each such candidate POMDP, we can compute the optimal policy for eventual comparison against the learned policy. We can potentially perform this computation using any existing solver, but we exploit an assumption of piecewise-linear models [Pynadath and Marsella, 2004] to arrive at a decision-tree representation of the optimal policy for both our candidate POMDPs and the learned Q values. We simply look for a POMDP whose decision-tree policy matches the decision-tree policy of the learned Q function. We then use that POMDP as the input to Section 3.2's templates, just as we did for the model-based RL's POMDP.

It is likely that there are multiple POMDPs consistent with the learned Q values, because although parameter changes will change the POMDP value function, most will not change the optimal policy. In such cases, we could simply pick one of the consistent POMDPs. It is an empirical question of how best to make this choice so that the agent provides explanation content that best calibrates trust with its human teammate.

Alternatively, we could focus on only the modeling components that overlap among the matching POMDPs. For example, if all the matching POMDPs have the same P function, but different O functions, then we would ignore P explanations and focus on O ones instead. Of course, it is possible that the ambiguity among the matching POMDPs extends to the individual components, in which case this method offers no advantage.

5.2 Learning-Based Explanation

The POMDP-based explanation from Section 3.2 reveals individual components of the agent's learned model. However, it does not inform the human teammate as to the overall content learned, nor how the learning arrived at that content. Fortunately, we can create additional templates for the components introduced by enabling our agent to use RL to update its model.

Q: The agent can explain its decisions by communicating the Q values that led to them. For example, the robot could say that "I currently estimate that not wearing protective gear is over 3 times better than wearing it."

$Q(b, \cdot)$: Alternatively, it could include the belief state on which the Q values are currently conditioned. For example, "Not wearing protective gear is over 3 times better than wearing it when there is a 90% chance that there is no threat."

$Q(\vec{\omega}, \cdot)$: As another alternative, the agent could describe its belief state in terms of the observations from which its current belief state is derived. For example, "Not wearing protective gear is over 3 times better than wearing it when none of my sensors detect any threats."

$\pi(b)$: While providing the Q values in the current belief state certainly helps transparency, the teammate might

benefit even more from understanding the agent's overall decision-making policy. For example, the robot could communicate context about the above examples by saying, "I recommend not wearing protective gear when I believe that there is a less than 25% chance of a threat." We leverage our piecewise-linear assumption to be able to generate such decision-tree representations of the agent's policy [Pynadath and Marsella, 2004].

$\pi(\vec{\omega})$: We can again formulate an alternative policy-based explanation using the sensor readings on which the relevant policy entry is contingent. For example, "I recommend not wearing protective gear whenever neither my NBC sensor nor microphone detect a threat."

Data: The Q values and policy certainly increase transparency, but they do not inform human teammates as how they were derived. It might therefore be useful to use the data used in the RL to arrive at the agent's current Q values. For example, "I recommend not wearing protective gear, because only 3 times (out of 91) was there actually a threat when none of my sensors did not detect any."

6 Discussion and Future Work

Just as with the POMDP-based explanation components of Section 3.2, it is an empirical question as to the impact of these different explanation forms on human-robot trust. The necessary next step is to conduct human-subject studies with our learning agent across different permutations of these explanations. Such an evaluation of the RL-based explanation can not only inform the efficacy of such explanations on building transparency, calibrating trust, and repairing trust, but also provide insight into what aspects of the RL should be included in the explanations and how to present them. For example, high-level explanations can provide at-a-glance information to help make decisions quickly, while detailed explanations can better help human teammates understand the agent's decision-making process. Human-interaction data can help evaluate such trade-offs and enable future agents to choose the right type of information to communicate at the right level of detail. More importantly, such evaluations can potentially reveal mechanisms to adapt to a human teammate's changing information and decision needs.

Another extension of the current work is to design bi-directional transparency communication that not only allows an agent to communicate its decision and explanations to its human teammate, but that also supports human input to the agent's decision-making, including interactive reinforcement learning similar to [Fukuchi *et al.*, 2017]. Such an investigation can also assist in more general classification of the individual differences that are prevalent in HRI domains [Pynadath *et al.*, 2018].

This paper's RL-based explanation mechanisms provide a simple way to systematically explore the impact of different XAI content on human trust perceptions and behavior in an HRI domain. Although most RL domains are more complex than our testbed domain, the POMDP and Q-learning components are identical. Therefore, we are hopeful that our findings about how different content affects human teammates will generalize to more realistic domains as well. As such, this paper outlines a potentially fruitful line of investigation into how best to automatically generate explanations of RL to benefit human-agent team performance.

Acknowledgments

This project is funded by the U.S. Army Research Laboratory. Statements and opinions expressed do not necessarily reflect the position or the policy of the United States Government, and no official endorsement should be inferred.

References

[Boutilier *et al.*, 2000] Craig Boutilier, Richard Dearden, and Moisés Goldszmidt. Stochastic dynamic programming with factored representations. *Artificial Intelligence*, 121(1):49–107, 2000.

[Dzindolet *et al.*, 2003] Mary T. Dzindolet, Scott A. Peterson, Regina A. Pomranky, Linda G. Pierce, and Hall P. Beck. The role of trust in automation reliance. *International Journal of Human-Computer Studies*, 58(6):697–718, 2003.

[Elizalde *et al.*, 2008] Francisco Elizalde, L. Enrique Sucar, Manuel Luque, J. Diez, and Alberto Reyes. Policy explanation in factored markov decision processes. In *Proceedings of the European Workshop on Probabilistic Graphical Models*, pages 97–104, 2008.

[Fukuchi *et al.*, 2017] Yosuke Fukuchi, Masahiko Osawa, Hiroshi Yamakawa, and Michita Imai. Autonomous self-explanation of behavior for interactive reinforcement learning agents. In *Proceedings of the 5th International Conference on Human Agent Interaction*, pages 97–101. ACM, 2017.

[Hayes and Shah, 2017] Bradley Hayes and Julie A Shah. Improving robot controller transparency through autonomous policy explanation. In *Proceedings of the 2017 International Conference on Human-Robot Interaction*, pages 303–312. ACM, 2017.

[Hendricks *et al.*, 2016] Lisa Anne Hendricks, Zeynep Akata, Marcus Rohrbach, Jeff Donahue, Bernt Schiele, and Trevor Darrell. Generating visual explanations. In *European Conference on Computer Vision*, pages 3–19. Springer, 2016.

[Iyer *et al.*, 2018] Rahul Iyer, Yuezhang Li, Huao Li, Michael Lewis, Ramitha Sundar, and Katia Sycara. Transparency and explanation in deep reinforcement learning neural networks. In *Proceedings of the Conference on Artificial Intelligence, Ethics, and Society*, 2018.

[Kaelbling *et al.*, 1996] Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

[Kaelbling *et al.*, 1998] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1):99–134, 1998.

[Kober *et al.*, 2013] Jens Kober, J Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.

[Lewicki, 2006] R. J. Lewicki. Trust, trust development, and trust repair. In M. Deutsch, P. T. Coleman, and E. C. Marcus, editors, *The handbook of conflict resolution: Theory and practice*, pages 92–119. Wiley Publishing, 2006.

[Matarić, 1997] Maja J. Matarić. Reinforcement learning in the multi-robot domain. *Autonomous Robots*, 4(1):73–83, 1997.

[Parasuraman and Riley, 1997] Raja Parasuraman and Victor Riley. Humans and automation: Use, misuse, disuse, abuse. *Human factors*, 39(2):230–253, 1997.

[Pynadath and Marsella, 2004] David V. Pynadath and Stacy C. Marsella. Fitting and compilation of multiagent models through piecewise linear functions. In *International Conference on Autonomous Agents and Multiagent Systems*, pages 1197–1204, 2004.

[Pynadath *et al.*, 2018] David V. Pynadath, Ning Wang, Ericka Rovira, and Michael J. Barnes. Clustering behavior to recognize subjective beliefs in human-agent teams. In *International Conference on Autonomous Agents and Multiagent Systems*, 2018.

[Ribeiro *et al.*, 2016] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM International Conference on Knowledge Discovery and Data Mining*, pages 1135–1144. ACM, 2016.

[Robinette *et al.*, 2015] Paul Robinette, Ayanna M. Howard, and Alan R. Wagner. Timing is key for robot trust repair. In *International Conference on Social Robotics*, pages 574–583. Springer, 2015.

[Russell and Norvig, 2016] Stuart J Russell and Peter Norvig. *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,, 2016.

[Schweitzer *et al.*, 2006] Maurice E. Schweitzer, John C. Hershey, and Eric T. Bradlow. Promises and lies: Restoring violated trust. *Organizational behavior and human decision processes*, 101(1):1–19, 2006.

[Sheh, 2017] Raymond Sheh. " why did you do that?" explainable intelligent robots. In *AAAI Workshop-Technical Report*, pages 628–634, 2017.

[Smart and Kaelbling, 2002] William D. Smart and Leslie Pack Kaelbling. Effective reinforcement learning for mobile robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 4, pages 3404–3410. IEEE, 2002.

[Sutton and Barto, 1998] Richard S. Sutton and Andrew G. Barto. *Reinforcement learning: An introduction*. MIT press Cambridge, 1998.

[Swartout and Moore, 1993] William R. Swartout and Johanna D. Moore. Explanation in second generation expert systems. In *Second generation expert systems*, pages 543–585. Springer, 1993.

[Swartout *et al.*, 1991] William Swartout, Cecile Paris, and Johanna Moore. Explanations in knowledge systems: Design for explainable expert systems. *IEEE Expert*, 6(3):58–64, 1991.

[Wang *et al.*, 2015] Ning Wang, David V. Pynadath, and Susan G. Hill. Building trust in a human-robot team. In *Interservice/Industry Training, Simulation and Education Conference*, 2015.

[Wang *et al.*, 2016] Ning Wang, David V. Pynadath, and Susan G. Hill. The impact of POMDP-generated explanations on trust and performance in human-robot teams. In *International Conference on Autonomous Agents and Multiagent Systems*, 2016.

[Wang *et al.*, 2018] Ning Wang, David V Pynadath, Ericka Rovira, Michael J Barnes, and Susan G Hill. Is it my looks? or something i said? the impact of explanations, embodiment, and expectations on trust and performance in human-robot teams. In *International Conference on Persuasive Technology*, pages 56–69. Springer, 2018.

[Zhang *et al.*, 2017] Yu Zhang, Sarath Sreedharan, Anagha Kulkarni, Tathagata Chakraborti, Hankz Hankui Zhuo, and Subbarao Kambhampati. Plan explicability and predictability for robot task planning. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 1313–1320. IEEE, 2017.

Analyzing Hypersensitive AI: Instability in Corporate-Scale Machine Learning

Michaela Regneri[◇], **Malte Hoffmann**[◇], **Jurij Kost**[◇], **Niklas Pietsch**[•], **Timo Schulz**[*], **Sabine Stamm**[•]
[◇]OTTO, [*]ITGAIN, [•]Otto Group Solution Provider
Hamburg, Germany
[◇]{firstname.lastname}@otto.de, [*]timo.schulz@itgain.de, [•]{firstname.lastname}@ottogroup.com

Abstract

Predictive geometric models deliver excellent results for many Machine Learning use cases. Despite their undoubted performance, neural predictive algorithms can show unexpected degrees of instability and variance, particularly when applied to large datasets. We present an approach to measure changes in geometric models with respect to both output consistency and topological stability. Considering the example of a recommender system using word2vec, we analyze the influence of single data points, approximation methods and parameter settings. Our findings can help to stabilize models where needed and to detect differences in informational value of data points on a large scale.

1 Introduction

Nowadays, Artificial Intelligences (AIs) beat humans in recognizing hand-written numbers [Alomari *et al.*, 2009], playing complex games like Go! [Silver *et al.*, 2017] and partially even in cancer diagnosis [Esteva *et al.*, 2017]. Many algorithms become more performant and more necessary than ever with the steady growth of available data. On the one hand, automated decisions become more accurate with more information to learn from, on the other hand, humans cannot keep pace with the velocity at which new data arises.

With the advance of neural networks and Big Data sources, results and mechanisms of Machine Learning components seem to have become less explainable. AI enables us to access useful data in the first place. However, the complexity of AI algorithms increases the gap between users as information consumers and the full background knowledge about the digested input's formation. AI consumers thus cannot always verify reliability and objectivity of automatically compiled content.

Instability in Machine Learning algorithms often exacerbates the problem of classifying AI output as trustworthy or questionable: The output of a learning algorithm might sometimes alter unexpectedly, even if the environment changes only marginally or not at all. Such instabilities can often be misinterpreted as unreliability of an algorithm, in particular for systems which require a high degree of guaranteed behaviour and reproducibility (e.g. a diagnostic assistant).

We take a step towards explaining output changes in geometric models in general, including neural models in particular. In an innovative experimental setup for sensitivity analysis, we apply topological and application-oriented metrics to detect differences between two recommender models. The experimental design disentangles two common sources of instability: We separate sensitivity that originates from minimal changes in the input data from variation originating from the algorithm itself, in particular from strategies that keep the approach compuationally tractable for large-scale applications.

Our main contribution is as follows: We present a combination of *scalable measures for sensitivity*, drawn from both, output consistency and topological model change. We apply those metrics to analyze word2vec [Mikolov *et al.*, 2013a], a widely used example for geometric models. As a result, we classify *factors of instability* originating from the algorithm itself and from properties of data points.

2 Related Work

While the algorithm in our focus was designed for Natural Language Processing, there are many non-linguistic applications of related vector models. For instance, skip-gram algorithms exist for different types of data, like social media graphs [Perozzi *et al.*, 2014], ordered products in e-commerce [Wang *et al.*, 2018], or click-through data as in our case.

Plenty of previous work is based on word2vec [Mikolov *et al.*, 2013a], and its results are analyzed thoroughly, but only few studies explain instabilities and the impact of randomness on the learned vector model. Wendlandt *et al.* [2018] show that even the embeddings of frequent words are unstable. Semantic properties of natural language (in particular, semantic relations) account for some of the sensitivities they find, which are thus hard to generalize to use cases that operate on other data sources rather than text.

Antoniak and Mimno [2018] show that inherent relations in the input data (like aforementioned semantic relations) can only partially explain the learned vector model. Some of the instability factors they find (like document length) might be use case generic. Other work [Caselles-Dupré *et al.*, 2018] highlights that instability can be also caused by default hyperparameters that are not tuned for a specific task.

One evaluation methodology we use resembles the lexical substitution task [McCarthy and Navigli, 2007], which is a popular benchmark for automatically generated semantic

similarity measures. Similar embedding methods to the one we employ have also been used for finding lexical substitutions [Melamud *et al.*, 2015, among others]. In context of recommender systems, the Netflix challenge [Bennett *et al.*, 2007] is a similar setting for predicting user-specific ratings.

Guss and Salakhutdinov [2018] derive a topological metric for measuring data complexity. One part of this metric is the number of the connected components, which we also employ as a metric. However, our goal differs from Guss and Salakhutdinov since they derive the appropriate neural architecture from the input data's topology, whereas we analyze the resulting embeddings.

3 Data and Algorithm

In this section, we briefly sketch recommender systems, introduce word2vec and describe our source data.

3.1 Recommender Systems

Recommender systems [Resnick and Varian, 1997] use automated means that help users make decisions with insufficient information. In general, such systems either use collaborative filtering, content-based recommendations or hybrid approaches. Collaborative filtering means that the (implicit or explicit) recommendations of all users are aggregated to find the best recommendations for the current user (such algorithms are mostly employed in web-shops for unknown users). Content-based recommendations refer to products previously bought or viewed by the current user and generate recommendations based on similarities with those products. Such similarities can be visual, textual or computed from other metadata (such as prices or brands).

Like many modern recommenders, the system we consider is a hybrid approach. Given a *seed product* (the product currently viewed by the user), we compute a sorted list of recommendations that are possible *alternatives* for the seed product (*"You might also like..."*). Such item-to-item recommendations are widely used in e-commerce. They increase conversion rates by helping customers find a product that meets their current need, and by keeping them engaged on the site. Technically, the alternatives in our model share similar click embeddings with the seed product, i.e. other customers looked at the alternative in similar user journey contexts in which they visited the current product in focus. For computing such embeddings, we use the word2vec algorithm.

3.2 Word2vec

Word2vec [Mikolov *et al.*, 2013a] is an algorithm that computes word embeddings. Words are represented by vectors, and the computed embeddings represent the words in their predicted contexts. In contrast to count-based co-occurrence models, word2vec and other predictive models use neural networks to compute a vector space, deriving abstract concepts as dimensions for each word from a given corpus. Count-based models define the vector space by directly using context words as dimensions, resulting in many more dimensions that are potentially less effective. Neural models perform superior in numerous tasks [Marco Baroni, 2014].

Word2vec applications comprise mostly tasks around Natural Language Processing, e.g. Sentiment Analysis [Liu, 2017], Information Retrieval [Onal *et al.*, 2017] and Named Entity Recognition [Das *et al.*, 2017], among many others. Neural predictions deliver good quality, which, however, comes at the cost of explainability. The features of word2vec are mostly opaque, and the resulting models change to a surprising extent even with only minor input variations [Wendlandt *et al.*, 2018]. The combination of high performance, frequent use, instability and unpredictability forges high interest in conditions that influence the algorithm, especially in corporate contexts.

3.3 Source Data

Our training data consists of anonymous user interactions with product pages. We define the chronologically sorted series of clicks by one user with breaks shorter than a certain time a *user session*. While a user session contains events of various types, we only consider clicks on product pages. Adopting linguistic terminology, we call one click in a session a *token*, and one abstract product, that can have multiple click instances, a *(product) type*. We use the term *product* interchangeably with *type*. The vector model we compute contains one vector per product type.

In our data model, each product belongs to a *product group* (e.g. the product group *coats* consists of several product instances, which can be parkas, jackets, or other coats). Those semantic groupings constitute a hierarchical system (all *coats* also belong to the more general group *clothing*).

In an operational recommender system, we train the recommender model on six months (180 days) of user sessions. Our standard training set of six months of user sessions contains about 98.4 million sessions with 1.3 million distinct products, of which we consider the 1.1 million products with a frequency of at least 5. There are 850 product groups, and each group contains 1,587.85 products on average. We update the system on a daily basis, which entails that the source data shifts by one day. While guaranteeing up-to-date recommendations and computational tractability, this method produces a permanent instability factor. Although we retain 99% of the data on each re-training, the results can differ considerably.

4 How to measure instability

We consider instability of an algorithm as the sum of unexpected changes, assessable in prediction changes of the resulting model. To measure instability, we thus need to be able to compare two models by quantifying and qualifying their differences. In the following, we describe the *metrics* to accomplish this kind of analyses, which are applied in the *experimental setup* outlined afterwards.

4.1 Metrics

We compare two click embeddings (trained on either two identical or modified data samples) using two types of metrics. One type is application-driven and complies directly with changes in recommendations. For this measurement, we measure the *neighborhood similarity* of the product vectors. The other type of metric assesses the geometric differences in the vector model itself and analyzes *the topological stability* of the manifolds constituted of the product vectors.

Neighborhood similarity: To compute neighborhood similarity of two vector models, we sample some vectors and compare their nearest neighbors according to cosine similarity in both models. The top *k* most similar vectors for a seed product represent the top *k* best recommendations for the seed product in our use case. For a given seed product *p* we define the top *k* overlap of two models as the fraction of the intersecting top *k* recommendations predicted by both models. In our experiment, we average over several pairwise model overlap scores, considering the intersection of a reference model and one test model at a time. For example, a top *15* overlap of 80% for a given seed product and a reference model means that on average, 12 of 15 recommendations predicted by the reference model are also predicted by the test models. Inversely, the same result quantifies instability as a difference of 20%. In our final evaluation, we compute the overlap for the 15 nearest neighbors, averaged over 5,000 seed products randomly sampled among the 10,000 most frequent products. While this excludes rare products, this sample reflects the instabilities we will mostly see in practice.

Topological stability: We complement the application-oriented neighborhood similarity with a model-driven measure which directly describes the geometric configuration of the vector model. As a tractable way to investigate properties and changes in vector constellation, we use a topological cluster analysis. Clusters arise from vectors according to the cosine similarity computed with the vector model. To determine the number of connected components, we use the DBScan clustering algorithm [Ester *et al.*, 1996]. The parameters for DBScan (neighborhood distance and the minimum number of neighbors) were heuristically determined and fixed for all runs. We require a core point to have at least ten neighbors with a cosine similarity > 0.8. For our purpose, we do not require an optimal clustering, we just need a setting that reflects model changes deterministically.

To measure topological stability, we compute the number of connected components that is equivalent to the number of clusters. Further, we analyze the size and purity of clusters as well as the total amount of noise.

Additionally, we measure *local density* in the vector space, counting the number of vectors within a certain distance from the clusters' centroids. We set this radius threshold to the value of ε-Parameter used for DBScan (0.8). To keep this analysis computationally tractable, we restrict the analysis to the 200 closest neighbors.

4.2 Experimental Setup

We want to identify and separate random factors from minimal changes in input data. In order to keep the algorithm computationally tractable, we compile a comparably small and representative data sample consisting of 1,309,907 user sessions (2 consecutive days). The sample contains 503,936 distinct product types. Ignoring products that occur less than 5 times (*–min-count 5*), the resulting word2vec output contains 253,889 product types. On average, one session consists of 6.93 product clicks, and the product tokens in one session belong to 2.26 distinct product groups.

The model computed from the whole sample dataset is our *reference model*. We analyze the impact of a single data point (cf. Sec. 5.3) with a leaving-one-out experiment. Based on the data sample described above, we create 500 subsamples by randomly leaving out one single session. For each subsample, we compute a vector model which we refer to as $model_{-1}$. The 500 sessions under consideration are representative with respect to session length (7.29 clicks).

We compute the product embeddings using Google's word2vec *WORD VECTOR estimation toolkit*, version 0.1c. For our sensitivity analysis towards minimal data changes, we need to eliminate algorithm-induced instabilities, which requires some parameter settings deviating from commonly used defaults. We choose skip-gram (*–cbow 0)*, no subsampling of frequent words (*–sample 0*), ten iterations (*–iter 10*) and only one thread due to thread-induced instabilities explained in Section 5.1. We stick to 100 dimensions (*–size 100*) and calculate the word vectors for all 501 samples in two runs: One using Hierarchical Softmax (later on referred to as HS with *–hs 1 –negative 0*) and one with negative sampling with 5 samples (later on referred to as NEG with *–hs 0 –negative 5*). Additionally, we fix the context window size to exactly 5 and round all vector numbers to 4 rather than 8 digits (the latter shows no impact in our results).

5 Factors of Instability

We analyze two types of instability. We consider the *instability due to random factors* and describe the *influence of the approximation method*. To disentangle artifacts from model training from actual sensitivity to input changes, we design an experiment that measures the prediction models' *instability due to minimal data changes*.

5.1 Random Factors out of the box

Using word2vec, we have learned that some seemingly random instabilities occur even for very simple, often small, datasets where, for instance, a single token that belongs to two sessions might introduce an opposing force that destabilizes the learned embedding. While our specific results are computed using Google's word2vec *WORD VECTOR estimation toolkit v 0.1c*, many findings probably carry over to other embedding algorithms, too.

Initially, we suspect random weight initialization in word2vec's code to account for most of the resulting variation. However, the word2vec implementation employs a number generator for each thread with a fixed seed and thereby guarantees reproducibility.

Analyzing other potential sources of randomness, we find one hyperparameter that induces a considerable amount of instability in the resulting word vectors: The *number of threads*.

Using multiple threads for training is vital for large datasets, allowing to distribute all data equally over all threads. Since all threads update the same weight matrix without locking, the resulting weight updates are non-deterministic. To verify the impact, we train five models on the same input file with 30 threads[1]. The models' agreement using similarity of the five nearest neighbors (cf. Sec. 4.1) results in an average overlap of 75% .

[1]Additional parameters: -window 5 -size 100 -sample 0 -min-count 5 -cbow 0 -hs 1 -negative 0 -iter 10

Another factor is *subsampling* that randomly omits single tokens from sessions with respect to the frequency of tokens, controlled by the hyperparameter *–sample*. Furthermore, the definition of the context window size (that is at max *–window*) depends on the same random numbers as the subsampling.

While the subsampling can be deactivated in v 0.1c, there is no hyperparameter to keep the window size constant. When keeping the input data identical, these two factors will have no "random" effect on the output vectors.

5.2 Approximation Methods for Model Training

We use an algorithm that computes word embeddings from sequential data for all types in a vocabulary. In their most basic count-based form, such embeddings treat each type as a dimension. With a growing vocabulary, training time increases considerably, because there are more vectors to be updated with each processed training token.

In order to keep embedding computation tractable and maintain the resulting predictions' quality, many algorithms use approximation methods, which vastly influence the learned vector model as well as its sensitivity towards changes in the training data. The word2vec implementation we use comes with the built-in options to either apply negative sampling (NEG), or Hierarchical Softmax (HS).

Negative Sampling: NEG saves runtime by reducing the number of vectors updated for each training token. The number is a new hyperparameter z. Instead of touching each vector in the vector space, only z vectors (and additionally the vector for the current training token) will change at a time. The vectors are chosen using some probability distribution D. D can, for instance, be set to be the unigram distribution, which ensures that vectors of frequent concepts are updated more often. Mikolov *et al.* show that this approach does not directly maximize the likelihood of correct tokens, but nevertheless it has been shown that this leads to useful embeddings and is thus a popular choice. In the implementation of word2vec that we use, z is by default set to 5.

Hierarchical Softmax: HS is inspired from Morin and Bengio (2005) and builds a binary tree to reduce the complexity from $O(n)$ to $O(log(n))$, with n being the number of types in the training corpus. Assuming e.g. one million of types, building the model requires only $log(n) \sim 20$ computation steps instead of $n = 10^6$ computations. The tree itself can be build in various ways. One common option is a Huffman tree that leads to binary codes where the length of each code is proportional to the frequency of the given type. In this tree, each node represents a unique type from the vocabulary with a unique code. Vector updates follow an order derived from root-to-leaf paths in the Huffman tree.

Choice of Approximation Method: Recent literature reflects NEG as the more popular option, but since more stable embeddings for recommendations are favored in practice, we nevertheless decide to use HS. We study the difference of both sampling methods by evaluating how the resulting embeddings react to minimal data change with the experiment outlined in Sec. 4. The correlation between the results of HS and NEG is about 50%. We find that NEG shows lower variance, but produces more sensitive models with respect to both environment and data changes.

Pr.	F1	F2	Code 1	Code 2	Dist.
A	15	15	01	10	2
B	10	10	00	00	0
C	9	9	110	110	0
D	8	8	101	011	2
E	6	6	1111	1111	0
F	4	4	1001	1110	3
G	4	3 (-1)	1110	0100	2
H	3	3	10001	01011	3
I	1	1	10000	01010	3

Table 1: An example assuming two datasets (1 & 2) with 9 products (Pr), their frequencies (F), Huffman codes & Hamming distances.

Hierarchical Softmax and Instabilities: For our experiments that require a deterministic approximation method, we modify HS's tree building mechanism by processing types in their lexical order whenever they have the same frequency.

While this modification guarantees identical prediction models for identical input, the tree building algorithm still accounts for much of the resulting models' sensitivity to seemingly inconsequential data changes: Given that our input data follows a Zipf distribution, many of our types have similar frequencies, especially the long tail of rare products. If one single token of a type t is omitted from the input (which means one click less), many leaves shuffle around and large parts of the inner tree structure can change. Each leaf node whose type shares t's original frequency and each type node with t's new frequency might change their position.

We illustrate such changes in the tree with a minimal example: Table 1 shows the Huffman codes (Code 1 and 2) arising from an artificial setting of 9 product types (Pr. A–I). We compare two settings (1 and 2) that differ only by one occurrence of G (in practice, this means a difference of one click). The Hamming distances ($Dist.$) between the two Huffman codes are shown in the last column. In this example, decrementing the frequency of one type by one results in an average Hamming distance of 1.67.

When we scale this experiment to our sample data with 1,309,907 million sessions and 253,889 different products, one might assume that leaving out one data point becomes a trivial change. Leaving out one session from 1,309,907 does, however, impact the Huffman coding considerably: Of the 500 subsamples that differ by one session, only 13 share the same set of Huffman codes (and thus the same tree) with the reference model. Within the 487 models that have a different Huffman coding, on average the codes of 46,244±59,015 products change in a range between 3 and 244,259 changed codes. As we will show in the next section, changes in the Huffman tree directly affect the resulting vector model, and thus the final predictions.

5.3 Results: Minimal Data Changes

We evaluate the impact of single data points (user sessions) in two ways. First, we analyze the influence on neighborhood

Feature	Description
Length	Number of products in session
Frequency	Average frequency in the vocabulary scaled by 1,000
Rank	Position in relative order in which the session is processed during training (scaled by 10^6)
Min Count	Boolean: Omitting the session means at least one type falls under min count threshold
Huffman Codes	Number of changed codes due to leaving out session (Log10)
Hamming Distance	Maximum change in binary code as quantified by Hamming distance

Table 2: Features for single data points (user sessions).

Feature	HS	NEG
Length	-0.077^{***} (0.004)	-0.072^{***} (0.005)
Frequency	0.248^{***} (0.059)	-0.487^{***} (0.091)
Huffman Codes	-1.200^{***} (0.077)	-0.305^{**} (0.120)
Hamming Distance	-0.160^{***} (0.019)	-0.057^{*} (0.030)
Min Count	0.239^{*} (0.142)	-1.470^{***} (0.222)
Rank	0.004 (0.009)	0.280^{***} (0.014)
Constant	96.100^{***} (0.279)	80.300^{***} (0.436)
Observations	500	500
Adjusted R^2	0.92	0.67

Level of significance: * p<0.1; **p<0.05; ***p<0.01

Table 3: Correlation coefficients of the linear regression model of data point features with the dependent variable: Overlap of 15 nearest neighbors. Standard Error in brackets.

similarities in detail (cf. Section 4.1) using a linear regression model. This experiment quantifies associations between particular data point characteristics and the degree of overlap to the reference model. Additionally, we analyze topological changes via cluster analysis, which indicates how the model shape reflects sensitivity to missing data points.

Data Point Features & Neighborhood Similarity

We compare the overlap of the 15 nearest neighbors for 5,000 vectors sampled from the 10,000 most frequent types. The average overlap is 93.6±2.65 for HS and 80.5±1.98 for NEG. With this measure as dependent variable, we analyze *stability* rather than instability, and hence draw the reverse conclusion. We consider a model sensitive or unstable, if omitting a data point results in a new $model_{-1}$ that has only a small overlap with the reference model. As sampling with frequent types might be suspected to induce a biased view on the overlap, we validate this assumption drawing a random sample of types. This decreases the level of overlap slightly, but due to an almost perfect correlation to the used variable, analytical relations are virtually unaffected.

Table 2 describes the explaining variables. From that set, session *Length* is the only property inherent to a data point, all other features describe data points in the training corpus' context (*Frequency*, *Rank*, *Min Count*). Associated with the sampling method, we also count the number of changes in the *Huffman codes* induced by leaving out this session (log10-scaled for a better fit), and the maximum change in binary code as quantified by the *Hamming distance*.

Our analysis also compares both approximation methods, Hierarchical Softmax (HS) and Negative Sampling (NEG), verifying results across both methods. Except for *Rank*, the features were selected such that the explanatory power for the HS-method is optimized; for the NEG-method choosing a different set of features would raise the degree of explicability. Table 3 summarizes the outcome: *Constant* shows the intercept of the regression model, *Observations* is the number of considered $model_{-1}$-experiments and *Adjusted* R^2 displays the adjusted coefficient of determination.

According to the regression model's results, the features account for about 92% of the variance in overlap (cf. Adjusted R^2) for HS, and about 67% for NEG. For the HS-model, the two measures reflecting changes in the Huffman tree are by far the most important ones. They alone explain 85% of the variation in the overlap, whereby stability is associated with fewer changes in the tree. A replicated leaving-one-out experiment with exactly the same Huffman tree for all runs shows this assumption to be true resulting in an average overlap of more than 99%.

Interestingly, both Huffman-related measures also show a weakly significant correlation with NEG-overlap, although Negative Sampling does not employ Huffman trees. Exploring this will be subject to future work.

Other relevant features include the session *Length*, which correlates negatively with the overlap. Its significance is, however, bounded compared to changes in the Huffman tree. In a univariate regression model, session length alone can explain about 35% of the variation. This relates to findings on linguistic data [Antoniak and Mimno, 2018], with both results confirming the expectation that larger data chunks (documents or sessions) contain more information.

Frequency is positively associated with overlap for HS. Omitting sessions with rare products decreases stability. Interestingly, rare items seem to affect NEG inversely.

Another feature, which has opposite associations for the two models, is *Min Count*. This binary variable indicates whether one or more products fall under the threshold for consideration. While it seems odd that the stability of HS increases if products are left out, the association is weakly significant.

One feature exclusively relevant for NEG is the position of the left-out session in the input data file (*Rank*). Sessions processed later have less influence on the models' stability.

Topological stability

We complement the analysis of neighborhood similarity with a study of model topologies. While the overlap of recommendations quantifies a change with respect to a reference model, topological figures might allow to directly benchmark models using their formal representations.

As an example, we illustrate the impact of a single data point on the number of clusters, which we calculate with the DBScan clustering algorithm (cf. Section 4.1). Figure 1

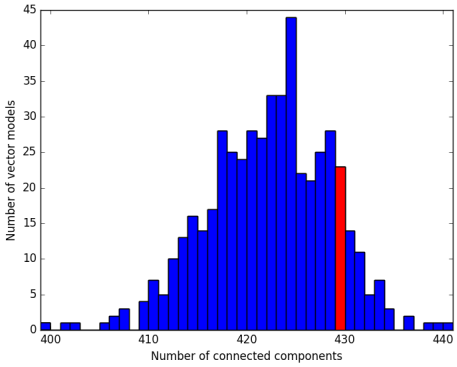


Figure 1: Number of clusters of the 501 click embeddings trained with HS. The bar with the reference model is marked in red.

shows a histogram of the 501 embeddings trained with the Hierarchical Softmax approximation. Each bar represents the number of $model_{-1}$-experiments (y-axis) resulting in the specific number of clusters (x-axis). The number of clusters ranges from 399 to 441 and the bar marked in red includes the reference model consisting of 429 clusters. The mean difference in the number of clusters with respect to the reference model is -7.23±6.21. In most cases, omitting a session reduces the number of clusters. Interestingly, the total number of noise points, which averages to 143,416±99, is virtually constant. Associated therewith, the average cluster size increases with a decreasing number of clusters, because the same number of points is divided into fewer clusters.

We also analyze the purity of clusters, which we define as the proportion of products belonging to the clusters' prevalent product group. Purity is thus minimal if each product in a cluster belongs to a different group, and maximal if all products in a cluster belong to the same group. The mean cluster purity for each embedding ranges from 0.903 to 0.917 with a mean of 0.910±0.002. The average purity depends on the number of small clusters, which contain fewer product groups and are thus purer in our experiment.

Tracing back the relationship of topological structure and model stability, we find mainly two bridges. First, differences in the number of clusters are significantly larger for two models with large maximum Hamming distances and many Huffman code changes, which is in line with the findings of our leaving-one-out experiment. Second, local density of the trained embeddings (cf. Section 4.1) correlates with the overlap measure. The larger the density of the embedding, the smaller the overlap. An intuitive explanation here is that there are many good recommendation candidates in a dense cluster, so the choice can vary more without sacrificing quality. Future research needs to show how exactly such topological changes relate to the omitted data points.

6 Conclusion

We showed an approach to sensitivity analysis of geometric predictive models, compiling metrics that measure both the topological stability of the vector model itself and the similarity of the output when changing parameters or leaving out single data points. As a basis for such an analysis, we identified and eliminated sources of randomness inherent to word2vec.

6.1 Main Contributions

Our main take-aways might help others to interpret and prevent instabilities in word2vec and similar algorithms:

- **Stable settings:** We identified sources of instability in word2vec and showed how to achieve constant results on identical input data. In particular, we showed that fixing the Huffman tree for HS can prevent variance even when the input data changes.
- **Choice of Approximation Method:** Choosing a different approximation method changes both, the result of a model and its sensitivity. In word2vec's case, Negative Sampling adopts new information more readily, while Hierarchical Softmax maintains more stability.
- **Data changes:** Even a single data point can change the models' output considerably. While the results' quality might be stable, the actual outputs change depending on the actual data points added or removed.
- **Information Density:** Data points, which hold more information, cause more changes in the final model. "More information" does not only refer to longer sentences or user sessions, but in particular to data points containing surprising associations of concepts, or information on rare concepts, or information on concepts for which the model learned a "cluttered" environment.

6.2 Discussion & Future Work

We presented a detailed way to measure model and output stability on an industrial scale. While certainly not comprehensive, our metrics can be a starting point for further analysis on model sensitivity of embeddings. We also believe that our methods and findings are applicable to linguistic and other applications. However, language as a data source behaves in a considerably different fashion than user interactions, thus possible differences concerning expressiveness and usefulness of our way to measure output consistency and quality can be expected. Comparing our results applied with linguistic experiments could show interesting differences.

In future work, we aim to additionally relate topological metrics to data point features. In the long run, we want to find a generic way to detect and predict sensitivity for all types of vector models, independently from particular use cases, data types or scale. Another goal is to highlight the relationship of model instability and business value of certain data points, assuming that data points which increase the model's output quality carry more valuable information. With such experiments, we could show not only data-point influence on model structure and predictions, but also measure the monetary value of model stability and the underlying data.

Acknowledgments

We want to thank Julia Soraya Georgi and two anonymous reviewers for their helpful comments. All remaining errors are of course our own.

References

[Alomari *et al.*, 2009] Saleh Alomari, Sumari Putra, Sadik Al-Taweel, and Anas J.A. Husain. Digital recognition using neural network. 5, 06 2009.

[Antoniak and Mimno, 2018] Maria Antoniak and David Mimno. Evaluating the stability of embedding-based word similarities. *Transactions of the Association for Computational Linguistics*, 6:107–119, 2018.

[Bennett *et al.*, 2007] James Bennett, Stan Lanning, and Netflix Netflix. The netflix prize. In *In KDD Cup and Workshop in conjunction with KDD*, 2007.

[Caselles-Dupré *et al.*, 2018] Hugo Caselles-Dupré, Florian Lesaint, and Jimena Royo-Letelier. Word2vec applied to recommendation: Hyperparameters matter. *CoRR*, abs/1804.04212, 2018.

[Das *et al.*, 2017] Arjun Das, Debasis Ganguly, and Utpal Garain. Named entity recognition with word embeddings and wikipedia categories for a low-resource language. *ACM Trans. Asian & Low-Resource Lang. Inf. Process.*, 16:18:1–18:19, 2017.

[Ester *et al.*, 1996] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, pages 226–231. AAAI Press, 1996.

[Esteva *et al.*, 2017] Andre Esteva, Brett Kuprel, Roberto A. Novoa, Justin Ko, Susan M. Swetter, Helen M. Blau, and Sebastian Thrun. Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, 542:115 EP –, 01 2017.

[Guss and Salakhutdinov, 2018] William H. Guss and Ruslan Salakhutdinov. On characterizing the capacity of neural networks using algebraic topology. *CoRR*, abs/1802.04443, 2018.

[Liu, 2017] Haixia Liu. Sentiment analysis of citations using word2vec. *CoRR*, abs/1704.00177, 2017.

[Marco Baroni, 2014] Germán Kruszewski Marco Baroni, Georgiana Dinu. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. *Proceedings of ACL 2014*, 1:238–247, 2014.

[McCarthy and Navigli, 2007] Diana McCarthy and Roberto Navigli. Semeval-2007 task 10: English lexical substitution task. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, SemEval '07, pages 48–53, Stroudsburg, PA, USA, 2007. Association for Computational Linguistics.

[Melamud *et al.*, 2015] Oren Melamud, Omer Levy, and Ido Dagan. A simple word embedding model for lexical substitution. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing, VS@NAACL-HLT 2015, June 5, 2015, Denver, Colorado, USA*, pages 1–7, 2015.

[Mikolov *et al.*, 2013a] Tomas Mikolov, Kai Chen, G.s Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. 2013, 01 2013.

[Mikolov *et al.*, 2013b] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc., 2013.

[Morin and Bengio, 2005] Frederic Morin and Yoshua Bengio. Hierarchical probabilistic neural network language model. In Robert G. Cowell and Zoubin Ghahramani, editors, *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, pages 246–252. Society for Artificial Intelligence and Statistics, 2005.

[Onal *et al.*, 2017] Kezban Dilek Onal, Ye Zhang, Ismail Sengor Altingovde, Md. Mustafizur Rahman, Pinar Senkul Karagoz, Alex Braylan, Brandon Dang, Heng-Lu Chang, Henna Kim, Quinten McNamara, Aaron Angert, Edward Banner, Vivek Khetan, Tyler McDonnell, An Thanh Nguyen, Dan Xu, Byron C. Wallace, Maarten de Rijke, and Matthew Lease. Neural information retrieval: at the end of the early years. *Information Retrieval Journal*, pages 1–72, 2017.

[Perozzi *et al.*, 2014] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 701–710, New York, NY, USA, 2014. ACM.

[Resnick and Varian, 1997] Paul Resnick and Hal R. Varian. Recommender systems. *Commun. ACM*, 40(3):56–58, March 1997.

[Silver *et al.*, 2017] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 10 2017.

[Wang *et al.*, 2018] Jizhe Wang, Pipei Huang, Huan Zhao, Zhibo Zhang, Binqiang Zhao, and Dik Lun Lee. Billion-scale commodity embedding for e-commerce recommendation in alibaba. *CoRR*, abs/1803.02349, 2018.

[Wendlandt *et al.*, 2018] L. Wendlandt, J. Kummerfeld, and R. Mihalcea. Factors influencing the surprising instability of word embeddings. *NAACL-HLT*, 2018.

A Survey of Interpretability and Explainability in Human-Agent Systems

Ariella Richardson[1], **Avi Rosenfeld**[1],
[1] Jerusalem College of Technology, Jerusalem 91160, Israel
rosenfa@jct.ac.il, richards@jct.ac.il

Abstract

This paper presents a taxonomy of interpretability in Human-Agent Systems. We consider four fundamental questions, "Why, what, when, and how" about interpretability. First, we consider why interpretability is needed in the system. Second, once interpretability is established as being needed, we consider what explanations can be generated to meet this need. Third, we consider when the user should be presented this information. Fourth, we consider the level of detail needed within explanations. Last, we consider how objective and subjective measures can be used to evaluate the entire system including the four fundamental questions regarding interpretability.

1 Introduction

As the field of Artificial Intelligence matures and becomes ubiquitous, there is a growing emergence of systems where people and agents work together. These systems, often called Human-Agent Systems or Human-Agent Cooperatives, have moved from theory to reality in the many forms, including digital personal assistants, recommendation systems, training and tutoring systems, service robots, natural language processing or chat bots, planning and self-driving cars [35; 39; 40; 41; 32; 23; 18; 30; 9; 26]. One key question surrounding these systems is the type and quality of the information that must be shared between the agents and the human-users during their interactions.

This paper focuses on one aspect of this human-agent interaction — the internal level of interpretability that the agents must present as potential explanations for their decisions within agents that use machine learning. Following previous work by Doshi-Velez and Kim, we refer to interpretability as the ability of an agent to explain or to present its decision to a human user, in understandable terms [8]. Given this definition, a system's explanation constitutes its interpretability and thus we will interchange use of both of these terms. This definition is intentionally left general to encompass the fields of Explainable AI (XAI) [13], where the explanations are geared towards people, and Interpretable Machine Learning [8] which primarily focuses on a machine learning algorithm's output. Despite an emerging interest in these fields, a limited number of works have addressed interpretability within human-agent systems.

We believe that the novelty within this paper is its ability to provide an interpretability framework that unifies elements of previous work [8; 10; 24; 13]. Specifically, we focus on four questions within Human-Agent Systems. We first address, "**Why** should a Human-Agent System be interpretable?" We propose a taxonomy of three reasons of interpretability: not-necessary (or marginally necessary), beneficial, and critical and discuss examples of each of these possibilities. This question must first be asked, as the its answer will likely impact the following issues. Second, we discuss, "**What** explanation can be generated?" We present three possible solutions for how to generate the information needed to make the system interpretable. At times, the explanations can be directly derived from the machine learning algorithm. In other cases, feature selection and/or analysis is used to generate explanations. Last, one might use a secondary explanation algorithm in addition to the machine learning algorithm. The third question, "**When** information should be presented?", addresses when the explanation should be presented: before, during or after the task is completed. We believe that the answer to the fourth question, "**How** much detail is presented in an explanation?" should be tied to the user's level of expertise and the type of system needed. The final issue we discuss is how explanations should be evaluated based on how well the human-agent system answers each of the previous four issues. We posit that both objective and subjective measures should be considered when evaluating such systems. Care must also be taken to address how these explanations may or may not have impacted the accuracy of machine learning algorithms and / or the user's productivity and satisfaction with the system.

Overall, we believe that the solutions presented to all of these issues need to be considered in tandem as they are intertwined. They directly depend on the type of human-agent system needing to be developed and thus directly stem from the first question about the overall reason, or reasons, that the system must be interpretable. Assuming that the system is human-centric, as is the case in recommendation [41], training [39], and tutoring systems [40], then the information will likely need to persuade the person to choose a certain action, for example through arguments about the agent's decision [27] or by using a suited presentation type [2]. If the system is agent-centric, such as in knowledge discovery or

self-driving cars, the agent might need to provide information about its decision to help convince the human participant of the correctness of their solution, aiding in the adoption of these technologies [25]. Furthermore, these explanations might be necessary for legal considerations [8]. In all cases we need to consider and then evaluate how these explanations were generated, presented, and if their level of detail correctly matches the system's need.

2 Why a Human-Agent System should be Interpretable

We posit that one can generalize the need for interpretability with a taxonomy of three levels:

1. Not helpful
2. Beneficial
3. Critical

Adjustable autonomy is a well-established concept within human-agent and human-robot groups that refers to the amount of control an agent/robot has compared to the human-user [11; 42; 31]. Under this approach, the need for interpretability can be viewed as a function of the degree of co-operation between the agent to the human user. Assuming the agent is fully controlled by the human operator (e.g. tele-operated), then no interpretability is needed as the agent is fully an extension of the human participant. Conversely, if the robot is given full control, particularly if the reason for the decision is obvious (a recommendation agent gives advice based on a well-established collaborative filtering algorithm), it again reasons that no interpretability is needed. Additionally, Doshi-Velez and Kim pointed out that explanation at times is not needed if there are no significant consequences for unacceptable results or the agent's decision is universally accepted and trusted [8].

At the other extreme, many Human-Agent systems are built whereby the agent's role is to support a human's task. In many of these cases, we argue that the agent's interpretation is a critical element within the system. For example, Intelligent Tutoring Systems (ITS) typically use step-based granularities of interaction whereby the agent confirms one skill has been learned or uses hints to guide the human participant [40]. The system must provide concrete explanations for its guidance (called *hints* in ITS terminology) to better guide the user. Similarly, explanations form a critical component of many negotiation, training, and argumentation human-agent systems [39; 35; 27; 29]. For example, explanations might be critical to aid a person in making the final life-or-death decision within human-agent systems [35]. Rosenfeld's et. al's NegoChat-A negotiation agent uses arguments to present the logic behind its position [29]. Traum et. al explained the justification within choices of their training agent to better convince the trainee, as well as teach the factors to look at in making decisions [39]. Rosenfeld and Kraus created agents that use argumentation to better persuade people to engage in positive behaviors, such as choosing healthier foods to eat [27]. Azaria et al. demonstrate how an agent that learns the best presentation method for proposals given to a user improves their acceptance rate [2]. Many of these systems can be generally described as Decision Support Systems (DSS). A DSS is typically defined as helping people make semi-structured decisions requiring some human judgment and at the same time with some agreement on the solution method [1]. An agent's effective explanation is critical within a DSS as the system's goal is providing the information to help facilitate improved user decisions.

A middle category in our taxonomy exists when interpretability is beneficial, but not critical. The Merrian-Webster dictionary defines beneficial as something that "produces good or helpful results"[1]. In general, we acknowledge that the value of interpretability can range widely between systems in this category. However, the defining characteristic is that the explanation provided is not inherently needed in order for the system to behave optimally or with peak efficiency. Instead, explanations in these cases can be helpful for **knowledge discovery** beyond the current primary objective of this system and might help further a second, but related goal. For example, a medical diagnostic system may work with peak efficiency exclusively as a *black box*. Nonetheless an agent that provides an explanation for its decision might further human understanding of a medical phenomenon. Along these lines, both the EU and UK governments have expressed a desire to confirm that AI algorithms are not biased against any ethic or gender groups [8]. Here again, the system's explanation is not critical for effective performance of the agent, but instead to confirm that a secondary, legal, requirement is being met. In fact, as we will further explore in the next section, an explainable model might even be done at a sacrifice to the system's performance.

Interpretability is sometimes beneficial to instill feelings of **trust and understanding** within the system's users. People are slow to adapt systems that they do not understand and trust: "If the users do not trust a model or a prediction they will not use it." [25]. Here, while the need for explanations may not arise as a result of the human-agent interaction (as it does in the extremes of full control by the agent or the human), yet these feelings are beneficial to help the human user work more effectively with the system or discard it when it is unreliable. To this end, Ribeiro et al. demonstrate how interpretability is important for identifying models that have high accuracy for the wrong reasons [25]. For example, they show that text classification often are wrongly based on the heading rather than the content. In contrast, image classifiers that capture the main part of the image in a similar manner to the human eye, install a feeling that the model is functioning correctly even if accuracy is not particularly high. Feelings of trust are especially important in certain domains, such as health-care [6], recommender systems [20], planning [9] and human-robot systems [30]. Furthermore, if the agent makes a mistake, the generated reasons could provide *transparency*, thus making it more likely the user will trust the agent in the future [9].

The benefit from this type of explanation will likely vary greatly across systems. If the user will not accept the system without this explanation, then a critical need for interpretabil-

[1] https://www.merriam-webster.com/dictionary/benefit

ity exists. However, in many, if not most, cases, the explanation is beneficial to the system's acceptance and / or to foster better trust. As the goal of interperability is different (e.g. improving user acceptance and trust instead of being vital to the success of the system), it reasons that the type of explanation may be fundamentally different. This in turn may impact the type of information the agent must present, something we address in Sections 4 and 5.

3 What Explanation can be Generated

Once we have established the **why** about explanations, a key related question one must address is **what** explanation can be generated. In addressing this point, we posit that three basic approaches exist as to how explanations can be generated:

1. Directly based on the machine learning algorithm
2. Feature selection and / or analysis
3. Using an explanation algorithm to create a visualization or explanation tool separate from the learning algorithm

The first approach, and possibly the most direct method, is to generate explanations directly from the output of the machine learning algorithm. For example, basic Case Based Reasoning algorithms will typically use k-nn algorithms to provide an explanation that two items are similar [37]. Decision trees have also been shown to be explainable, especially if the size of the tree is limited and have already been established for their ability to explain the model in real-world systems [10; 13; 28]. These two algorithms are potentially clearly explainable machine learning algorithms that could be used to provide critically important levels of explanation. A clear downside to this approach is that one is then limited to a specific algorithm, or a specific algorithmic implementation, that provides the necessary explanations, even when it is less accurate than other alternatives. It has been previously noted that an inverse relationship often exists between machine learning algorithms' accuracy and their explainability [13]. *Black box* algorithms, especially deep neural networks, are often used due to their exceptional accuracy on some problems. However, these types of algorithms are difficult to glean explanations from. Figure 1 is based on previous work [13; 7] and quantifies the general relationship between algorithms' explainability and accuracy. One possible solution to achieve both high accuracy and explainability is to create new machine learning algorithms that explicitly consider the amount of explanation needed by the system. One example of this approach is work by Kim, Rudin and Shah who have suggested a Bayesian Case Model for case-based reasoning [19].

A second approach is to perform feature selection and / or feature analysis before or after a model has been built. Feature selection has long been established as an effective way of building potentially better models which are simpler and thus better overcome the curse of dimensionality [14]. Additionally, models with fewer attributes are potentially easier to understand as the true causal relationship between the dependent and independent variables is clearer and thus easier to present to the user [21]. The strong advantage of this approach is that the information presented to the user is generated directly from the mathematical relationship between a small set of features and the target being learned. However, such approaches are inherently limited if models generated by this approach are not sufficiently accurate and / or significantly better models can be generated by less explainable models (e.g. deep learning). Furthermore, even methods that are typically more understandable, such as decision trees, are less interpretable if the size of the tree is too large to fully understand (e.g. thousands of rules) [15].

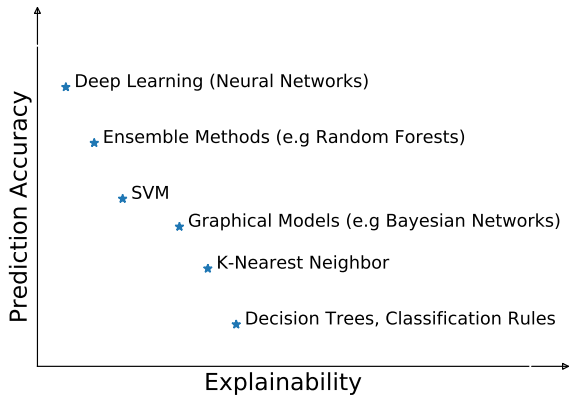


Figure 1: : Prediction accuracy versus explainability

A third approach is to use an algorithm in addition to the agent's machine learning and then use this information to provide a visualization or text-based explanation. This type of approach can be defined as a metacognition process [5], or the process of reasoning about the machine learning process. One example of theses algorithms is LIME (Local Interpretable Model-Agnostic Explanations), which provides explanations without any connection to the type of machine learning algorithm used [25]. It is noteworthy that LIME includes feature engineering as part of its analysis, showing the potential connection between the second and third approaches. A second possibility includes algorithms that provide visualization tools as explanations. For example, saliency maps can provide a visualization tool that is effective in better understanding image classification in deep networks [36]. Visualization of the algorithm output, when possible, is also beneficial to interpretability, such as visualizing a decision tree [32]. A third possibility focuses on developing explanations of algorithms through text explanations. For example, such tools have been suggested for generating explanations of deep networks for image classification [17].

Legal and practical considerations might limit researchers as to what constitutes a sufficient explanation, especially if only certain algorithms provide the critically needed level of interpretability (see Section 2). This in turn impacts what type of explanation approach can be implemented. Additionally, the decision on **when** this information is presented can often be connected to the type of explanation that has been generated, something we further explore in the next section.

4 When Should Information be Presented

Interpretations can be categorized based on when the explanation is made:

1. Before the task
2. Continued explanations during the task
3. After the task

Some agents may present their explanation **before** the task is executed as either justification, conceptualization or explanation for an agent's plan of action. Other agents may present their explanation **during** task execution, especially if this information is important to explain when the agent fails so it will be trusted to correct the error. Others agents provide explanations **after** actions are carried out [23], to be used for retrospective reports or post-hoc analysis [24].

The choice of when to present the explanation is not exclusive. Agents might supply various explanations at various times, before, during and after the task is carried out. Building on the taxonomy in the previous section, if interpretability is critical to the system, then it stands to reason that this knowledge must be presented at the **beginning** of the task, thus enabling the user to determine whether to accept the agents recommendation [32]. However, if it is beneficial to build trust / user acceptance, then it might be directed **during** the task, especially if the agent erred. If the purpose of the explanation is to justify the agent's choice from a legal perspective then we may need to certify that decision before the agent acts (preventative) or after the act (accusatory). But, if the goal is conceptualization, especially in the form of knowledge discovery and / or to support future decisions, then the need for explanation **after** task execution is equally critical.

5 How Much Detail is Presented in an Explanation

The level of explanation detail should depend on why that human-agent systems needs to be interpretable and how the explanation has been generated. If the need for explanation is for legal purposes, then it follows that legal experts need the explanation and not the typically user. Similarly, it reasons that the type of explanation that is given should be directed specifically to this population. If the purpose the explanation is to support expert knowledge discovery, then it reasons that the explanation should be directed towards researchers with knowledge of a specific problem. In these cases, the systems might not even need to present their explanations to the *regular users* and may thus only focus on presenting information to these experts. Most systems will still likely benefit by directing explanation to the *regular users* to help them better understand the system's decisions, thus aiding in their acceptance and / or trust. In these cases, the system should be focused on providing arguments and / or the logic behind their decisions [27] or Case Based Reasoning (e.g. I choose to do this before of that) [22; 4; 19] that help reassure the user about the correctness of the agent's decision.

Similarly, the level of detail of what constitutes an adequate explanation likely depends on the project amount of time the user will study the explanation. If the goal is knowledge discovery and / or legal, then an expert will likely need to spend large amounts of time meticulously studying the inner-workings of the decision making process. In these cases, it seems likely that large amounts of detail regarding the justification of the explanations will be necessary. If a *regular user* is assumed, and the goal is to build user trust and understanding, then shorter, very directed explanations are likely more beneficial. This issues touches upon a larger issues about the potential information overload additional information may pose for a given user [34].

Additionally, the type of interface used for disseminated the system's explanation will likely depend upon the level of the user's expertise. The idea of adaptable interfaces based on people's expertise was previously noted [12; 33; 38]. In these systems, the type of information presented in the interface depends on the user's level of expertise. Accordingly, an interface might consider different types of explanations or explanation algorithms based on who the end-user will be. Consider explanations that can potentially be generated within an image classification task. Most people will be able to understand the comparison of similar pictures or even basic text explanations that have been previously proposed [17]. However, interfaces with saliency maps [36] may be too complex for many users and thus reserved for experts, especially those researching the under-workings of the algorithm. Even among experts, it is reasonable to assume that different users will need different types of information. The different backgrounds of legal experts, scientists, safety engineers, and researchers may necessitate different types of interfaces [8].

6 A Utility for Evaluation of an Explanation Based System

We previously considered issues of **Why**, **What** ,**When** and **How** regarding explanations. These issues are often intrinsically connected. For example, the detail of an explanation is often dependent on **why** that explanation is needed. An expert will likely differ from a *regular user* regarding **why** an explanation is needed, will often need these explanations at different times, e.g. before or after the task (**when**), and may require different types of explanations and interfaces (**what** and **how**). At other times multiple facets of explanation exist even within one category. A DSS system is built to support a user's decision, thus making interpretability a critical issue. However, these systems will still likely benefit from better explanations, so that the user trusts those explanations. Similarly, a scientist pursuing knowledge discovery may need to analyze and interact with information presented before, during and after a task's completion (**when**). Thus, multiple goals must often be considered and evaluated.

We consider three elements that should be considered in evaluating Human-Agent Systems:

1. A score for the performance of the agent's algorithm
2. A score for the user's performance
3. A score for the explanation given the to user

For example, in a movie recommendation system the three scores would be described as follows: The score of the algorithmic component is an evaluation of the algorithm that is used for recommendation (i.e. accuracy, precision and / or recall). The user's performance score evaluates whether the

user chooses more movies when using the system. The score of the explanation refers to user's satisfaction.

A complex interplay exists between these three elements. Both the user's performance and the user's satisfaction can be affected by the explanation. On the other hand they are also both affected by the algorithm. As we discussed in Section 3, there is often a trade-off between the performance of the agent and the ability to provide a high quality explanation. The information generated from simpler machine learning algorithms might be more explainable, but the accuracy of these algorithms might be lower than those of less explainable machine learning algorithms (see Figure 1). Furthermore, different user types and interfaces will be effected by the type of agent design and a total measure is needed to weigh all parameters that are needed by a system into account. For example, an agent that was designed to support an *expert user* is different to that provided to a *regular user*.

We must consider situations when the goals for system are both complementary and in conflict. Earlier, we discussed several instances where the goals are complementary. However, this is not always the case. For example, assuming the explanation goal of a system is to support a person's ability to purchase items in a time-constrained environment (e.g. online stock purchasing). The greater detail contained within the agent's explanations on one hand instill improved confidence within the user, but also will take more time to read and process, which may prevent the user from capitalizing on certain quickly passing market fluctuations. Thus, some measure should likely be introduced to reason about different goals for the explanation and the relative strengths of various explanations, their interfaces, and the algorithms that generate those explanations.

Another equally important element of the system is how well the person performed in the system with the total of all explanations provided. In theory, different such goals may exist for the human user such as immediate performance vs. long-term knowledge acquisition. Furthermore, the performance of the agent's algorithm is an important part of the system and must be included in the system utility, regardless of which specific user or user constraints exist. Clearly, we need to maximize the overall utility of the system across all elements of the system's performance.

To capture these properties, we propose an overall utility to the system which is the following weighted sum:

$$Utility = \sum_{n=1}^{NumGoals} Imp_n * Grade_n \quad (1)$$

We define $NumGoals$ as the number of goals in an the system. Examples of goals could be "explanation during agent activity", "explanation after completion of task or "clarity of visualization for explanation" or "clarity of visualization for explanation". Goals can also refer to the system performance, such as "user performance". A goal can also be related to the algorithmic component, such as "model accuracy". Imp_n is the importance weight we give to the n_{th} goal. Similarly, $Grade_n$ is the score we give to the n_{th} goal. We require that $0 <= Grade_n <= 1$ such that

$$\sum_{n=1}^{NumGoals} Imp_n = 1 \quad (2)$$

While this model helps quantify the interplay of multiple explanation goals, either inherently complimentary or contradictory, a fundamental question lingers about how to set the values of $NumGoals$, Imp_n and $Grade_n$. We assume that either users themselves the system's designer, or outside 3rd party organizations (e.g. governments) can quantify both the goals of the system and their relative importance. The grade for the user's performance is assumed to be quantifiable. Assuming the goal is to support the user's ability to classify an item, then accepted measures such as accuracy, precision, recall, F-measure, and Mean Absolute Errors can be used too. However, it reasons that the score of a given goal may also be boolean (e.g. either the system provided an acceptable legal explanation or it didn't) while others are highly subjective and may vary as the task is performed (e.g. how much trust did the agent generate). If providing a certain explanation is a critically important goal, then this value should be made much larger than any of the other goals to ensure its primacy.

According to this model, some measure of the user's performance with the system is needed to quantify $Grade_n$ for this goal. Assuming the user experience needs to be measured, as is typically done in HCI studies and is likely to be a significant factor in such human-agent systems, then accepted user performance metrics such as the NASA-TLX [16] or the System Usability Scale [3] should be used to measure the utility of the user's satisfaction. In all cases, we concede that for many real-world systems quantifying these elements, especially in the case of the relative values for each $Grade_n$ is an important challenge that needs to be further explored in the future.

In order to help evaluate these systems, some researchers have suggested simplifying the type of domains, experiment setups and evaluation criteria. One possible approach is to objectively quantify the grade for the explanation portion of an agent's model based on its size. The assumption behind this approach is that as the size of machine learning models grow, they are less interpretable. Thus, models with large numbers of nodes / hidden layers (e.g. in deep neural networks), parameter values (for regression and SVM models), the number of rules (rule-based models), or the depth (in decision trees) are less preferable to those with fewer numbers of these values [7]. Following Section 4, we could also create an objective metric based on the number of features generated from feature selection that are used to create the model. The advantage to both of these approaches is the the value of explanation's $Grade_n$ can be made independently to the system's specific task. A second approach is to quantify the relatively value of different approaches and only give a non-zero score to the one that they feel is best [8]. Thus, the scoring function $Grade_n$ could be made boolean, greatly simplifying calculating the system's total value. A third approach is to create simplified accepted tasks, potentially where simulations of human behavior could be used for repeatedly for evaluation across different algorithms, interfaces, and approaches [8]. This could create a standardization for all values of

$NumGoals$, Imp_n and $Grade_n$, again greatly aiding in the evaluation process. Currently, no such canonical tasks have been universally accepted, leaving this issue as an open challenge.

7 Conclusion

We presented a framework designed to enable comparison and evaluation of interpretability in Human-Agent Systems. As Human-Agent Systems are diverse and complex, there is no "one explanation type fits all". Each agent must have its requirements and goals mapped out, and the appropriate explanation chosen. We focused on agents that use machine learning and provided an attempt to define this new field of interpretability and explainability.

Our contribution is a proposed framework that determines the answers to four questions **Why**, **What**, **When** and **How**. These questions define the various aspects of the explanation for the system. In designing an agent one must first establish **why** the system requires explanation, as this will affect the answer to the other questions. Next, one must determine **what** type of explanation is needed, followed by considering **when** to present it. Finally, the question of **how** detailed the explanation should be must be addressed. For each of the first three questions we presented a set of three possible approaches, and discussed when each approach might be appropriate. Various factors affect the answers to these questions. We discussed how the degree of control of the user over the agent affects the need for interpretability. We also discussed how the type of learning that agents perform will affect the explanation that is provided. We then discussed parameters for when to present the information. For the fourth question of **how** much detail to present, we consider a continuous scale, and discuss the different types of detail needed in different systems, and how the answers to the first three questions affect this decision.

Once an explanation has been defined, there is a need to evaluate it. To this end, we presented an evaluation measure. The measure allows for comparing systems while taking into account both the algorithmic performance and the presentation of explanations in order to achieve the highest interpretability and performance. Our proposed utility is capable of combining all the aspects of the system: the machine learning algorithm, user performance and the explanation, into a single measure. We discussed the strengths and limitations of our proposed measure. While the measure provides a means for comparison, its main limitation relates to the elements that can potentially be subjective in determining the values of the parameters. As this remains an open issue, we hope that this unified framework will not only be adopted by researchers when defining and evaluating the important field of interpretability in Human-Agent Systems, but will provide a basis for possible extensions.

References

[1] Frederic Adam, Frederic Adam, and Patrick Humphreys. *Encyclopedia of Decision Making and Decision Support Technologies*. Information Science Reference - Imprint of: IGI Publishing, Hershey, PA, 2008.

[2] Amos Azaria, Ariella Richardson, and Sarit Kraus. An agent for the prospect presentation problem. In *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*, pages 989–996. International Foundation for Autonomous Agents and Multiagent Systems, 2014.

[3] John Brooke et al. Sus-a quick and dirty usability scale. *Usability evaluation in industry*, 189(194):4–7, 1996.

[4] Juan M Corchado and Rosalía Laza. Constructing deliberative agents with case-based reasoning technology. *International Journal of Intelligent Systems*, 18(12):1227–1241, 2003.

[5] Michael T Cox and Anita Raja. *Metareasoning: Thinking about thinking*. MIT Press, 2011.

[6] David Crockett and Brian Eliason. What is data mining in healthcare?, 2016.

[7] Hoa Khanh Dam, Truyen Tran, and Aditya Ghose. Explainable software analytics. *CoRR*, abs/1802.00603, 2018.

[8] Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. 2017.

[9] Maria Fox, Derek Long, and Daniele Magazzeni. Explainable planning. *CoRR*, abs/1709.10256, 2017.

[10] Alex A. Freitas. Comprehensible classification models: A position paper. *SIGKDD Explor. Newsl.*, 15(1):1–10, March 2014.

[11] Michael A Goodrich, Dan R Olsen, Jacob W Crandall, and Thomas J Palmer. Experiments in adjustable autonomy. In *Proceedings of IJCAI Workshop on Autonomy, Delegation and Control: Interacting with Intelligent Agents*, pages 1624–1629. Seattle, WA: American Association for Artificial Intelligence Press, 2001.

[12] Jonathan Grudin. The case against user interface consistency. *Communications of the ACM*, 32(10):1164–1173, 1989.

[13] David Gunning. Explainable artificial intelligence (xai). *Defense Advanced Research Projects Agency (DARPA)*, 2017.

[14] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar):1157–1182, 2003.

[15] Satoshi Hara and Kohei Hayashi. Making tree ensembles interpretable. *arXiv preprint arXiv:1606.05390*, 2016.

[16] Sandra G Hart and Lowell E Staveland. Development of nasa-tlx (task load index): Results of empirical and theoretical research. In *Advances in psychology*, volume 52, pages 139–183. Elsevier, 1988.

[17] Lisa Anne Hendricks, Zeynep Akata, Marcus Rohrbach, Jeff Donahue, Bernt Schiele, and Trevor Darrell. Generating visual explanations. In *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part IV*, pages 3–19, 2016.

[18] Nicholas R Jennings, Luc Moreau, David Nicholson, Sarvapali Ramchurn, Stephen Roberts, Tom Rodden, and Alex Rogers. Human-agent collectives. *Communications of the ACM*, 57(12):80–88, 2014.

[19] Been Kim, Cynthia Rudin, and Julie A Shah. The bayesian case model: A generative approach for case-based reasoning and prototype classification. In *Advances in Neural Information Processing Systems*, pages 1952–1960, 2014.

[20] Bart P Knijnenburg, Martijn C Willemsen, Zeno Gantner, Hakan Soncu, and Chris Newell. Explaining the user experience of recommender systems. *User Modeling and User-Adapted Interaction*, 22(4-5):441–504, 2012.

[21] Igor Kononenko. Explaining classifications for individual instances. In *In Proceedings of IJCAI'99*, pages 722–726, 1999.

[22] Oh Byung Kwon and Norman Sadeh. Applying case-based reasoning and multi-agent intelligent system to context-aware comparative shopping. *Decision Support Systems*, 37(2):199–213, 2004.

[23] Pat Langley, Ben Meadows, Mohan Sridharan, and Dongkyu Choi. Explainable agency for intelligent autonomous systems. In *AAAI*, pages 4762–4764, 2017.

[24] Zachary Chase Lipton. The mythos of model interpretability. *CoRR*, abs/1606.03490, 2016.

[25] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1135–1144. ACM, 2016.

[26] Ariella Richardson, Sarit Kraus, Patrice L Weiss, and Sara Rosenblum. Coach-cumulative online algorithm for classification of handwriting deficiencies. In *AAAI*, pages 1725–1730, 2008.

[27] Ariel Rosenfeld and Sarit Kraus. Strategical argumentative agent for human persuasion. In *ECAI*, volume 16, pages 320–329, 2016.

[28] Avi Rosenfeld, Vinay Sehgal, David G. Graham, Matthew R. Banks, Rehan J. Haidry, and Laurence B. Lovat. Using data mining to help detect dysplasia: Extended abstract. In *2014 IEEE International Conference on Software Science, Technology and Engineering, SWSTE 2014, Ramat Gan, Israel, June 11-12, 2014*, pages 65–66, 2014.

[29] Avi Rosenfeld, Inon Zuckerman, Erel Segal-Halevi, Osnat Drein, and Sarit Kraus. Negochat-a: a chat-based negotiation agent with bounded rationality. *Autonomous Agents and Multi-Agent Systems*, 30(1):60–81, 2016.

[30] Maha Salem, Gabriella Lakatos, Farshid Amirabdollahian, and Kerstin Dautenhahn. Would you trust a (faulty) robot?: Effects of error, task type and personality on human-robot cooperation and trust. In *Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction*, pages 141–148. ACM, 2015.

[31] Paul Scerri, David Pynadath, and Milind Tambe. Adjustable autonomy in real-world multi-agent environments. In *Proceedings of the fifth international conference on Autonomous agents*, pages 300–307. ACM, 2001.

[32] Raymond Sheh. why did you do that?" explainable intelligent robots. In *AAAI Workshop on Human-Aware Artificial Intelligence*, 2017.

[33] Ben Shneiderman. Promoting universal usability with multi-layer interface design. *ACM SIGCAPH Computers and the Physically Handicapped*, (73-74):1–8, 2002.

[34] Tammar Shrot, Avi Rosenfeld, Jennifer Golbeck, and Sarit Kraus. Crisp: an interruption management algorithm based on collaborative filtering. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 3035–3044. ACM, 2014.

[35] Maarten Sierhuis, Jeffrey M Bradshaw, Alessandro Acquisti, Ron Van Hoof, Renia Jeffers, and Andrzej Uszok. Human-agent teamwork and adjustable autonomy in practice. In *Proceedings of the seventh international symposium on artificial intelligence, robotics and automation in space (I-SAIRAS)*, 2003.

[36] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *CoRR*, abs/1312.6034, 2013.

[37] Frode Sørmo, Jörg Cassens, and Agnar Aamodt. Explanation in case-based reasoning—perspectives and goals. *Artif. Intell. Rev.*, 24(2):109–143, October 2005.

[38] Sebastian Stein, Enrico H. Gerding, Adrian Nedea, Avi Rosenfeld, and Nicholas R. Jennings. Market interfaces for electric vehicle charging. *J. Artif. Intell. Res.*, 59:175–227, 2017.

[39] David Traum, Jeff Rickel, Jonathan Gratch, and Stacy Marsella. Negotiation over tasks in hybrid human-agent teams for simulation-based training. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 441–448. ACM, 2003.

[40] Kurt VanLehn. The relative effectiveness of human tutoring, intelligent tutoring systems, and other tutoring systems. *Educational Psychologist*, 46(4):197–221, 2011.

[41] Bo Xiao and Izak Benbasat. E-commerce product recommendation agents: use, characteristics, and impact. *MIS quarterly*, 31(1):137–209, 2007.

[42] Holly A Yanco and Jill Drury. Classifying human-robot interaction: an updated taxonomy. In *Systems, Man and Cybernetics, 2004 IEEE International Conference on*, volume 3, pages 2841–2846. IEEE, 2004.

A Symbolic Approach to Explaining Bayesian Network Classifiers*

Andy Shih and **Arthur Choi** and **Adnan Darwiche**
Computer Science Department
University of California, Los Angeles
{andyshih,aychoi,darwiche}@cs.ucla.edu

Abstract

We propose an approach for explaining Bayesian network classifiers, which is based on compiling such classifiers into decision functions that have a tractable and symbolic form. We introduce two types of explanations for why a classifier may have classified an instance positively or negatively and suggest algorithms for computing these explanations. The first type of explanation identifies a minimal set of the currently active features that is responsible for the current classification, while the second type of explanation identifies a minimal set of features whose current state (active or not) is sufficient for the classification. We consider in particular the compilation of Naive and Latent-Tree Bayesian network classifiers into Ordered Decision Diagrams (ODDs), providing a context for evaluating our proposal using case studies and experiments based on classifiers from the literature.

1 Introduction

Recent progress in artificial intelligence and the increased deployment of AI systems have led to highlighting the need for *explaining* the decisions made by such systems, particularly classifiers; see, e.g., [Ribeiro *et al.*, 2016b; Elenberg *et al.*, 2017; Lundberg and Lee, 2017; Ribeiro *et al.*, 2018].[1] For example, one may want to explain *why* a classifier decided to turn down a loan application, or rejected an applicant for an academic program, or recommended surgery for a patient. Answering such *why?* questions is particularly central to assigning blame and responsibility, which lies at the heart of legal systems and may be required in certain contexts.[2]

In this paper, we propose a *symbolic* approach to explaining Bayesian network classifiers, which is based on the following observation. Consider a classifier that labels a given instance either positively or negatively based on a number of discrete features. Regardless of how this classifier is implemented, e.g., using a Bayesian network, it does specify a symbolic function that maps features into a yes/no decision (yes for a positive instance). We refer to this function as the classifier's *decision function* since it unambiguously describes the classifier's behavior, independently of how the classifier is implemented. Our goal is then to obtain a symbolic and tractable representation of this decision function, to enable symbolic and efficient reasoning about its behavior, including the generation of explanations for its decisions. In fact, [Chan and Darwiche, 2003] showed how to compile the decision functions of naive Bayes classifiers into a specific symbolic and tractable representation, known as Ordered Decision Diagrams (ODDs). This representation extends Ordered Binary Decision Diagrams (OBDDs) to use multi-valued variables (discrete features), while maintaining the tractability and properties of OBDD [Bryant, 1986; Meinel and Theobald, 1998; Wegener, 2000].

We show in this paper how compiling decision functions into ODDs can facilitate the efficient explanation of classifiers and propose two types of explanations for this purpose.

The first class of explanations we consider are *minimum-cardinality explanations*. To motivate these explanations, consider a classifier that has diagnosed a patient with some disease based on some observed test results, some of which were positive and others negative. Some of the positive test results may not be necessary for the classifier's decision: the decision would remain intact if these test results were negative. A minimum-cardinality explanation then tells us which of the positive test results are the culprits for the classifier's decision, i.e., a minimal subset of the positive test results that is sufficient for the current decision.

The second class of explanations we consider are *prime-implicant explanations*. These explanations answer the following question: what is the smallest subset of features that renders the remaining features irrelevant to the current decision? In other words, which subset of features—when fixed—would allow us to arbitrarily toggle the values of other features, while maintaining the classifier's decision?

This paper is structured as follows. In Section 2, we review the compilation of naive Bayes classifiers into ODDs, and propose a new algorithm for compiling latent-tree classifiers into ODDs. In Section 3, we introduce minimum-cardinality

*This is an abridged version of a paper with the same title, to be presented at IJCAI-ECAI 2018.

[1] It is now recognized that opacity, or lack of explainability is "one of the biggest obstacles to widespread adoption of artificial intelligence" (The Wall Street Journal, August 10, 2017).

[2] See, for example, the EU general data protection regulation, which has a provision relating to explainability, https://www.privacy-regulation.eu/en/r71.htm.

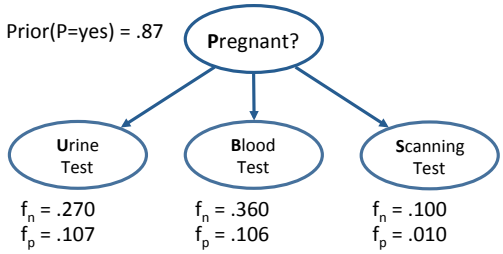


Figure 1: A naive Bayes classifier, specified using the class prior, in addition to the false positive (f_p) and false negative (f_n) rates of features. The class variable and features are all binary.

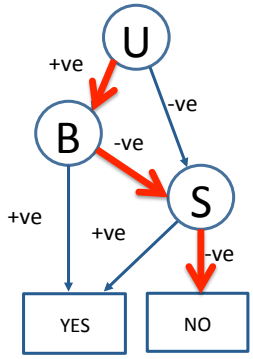


Figure 2: An OBDD (decision function) of the classifier in Figure 1.

explanations, propose an algorithm for computing them, and provide a case study on a real-world classifier. In Section 4, we do the same for prime-implicant explanations. In Section 5, we discuss the relationship between the two types of explanations and show that they coincide for monotone classifiers. We then follow by a discussion of related work in Section 6 and finally close in Section 7. Proofs of theorems appear in the full version of this paper [Shih *et al.*, 2018].[3]

2 Compiling Bayesian Network Classifiers

Consider Figure 1 which depicts a naive Bayes classifier for detecting pregnancy. Given results for the three tests, if the probability of pregnancy passes a given threshold (say 90%), we would then obtain a "yes" decision on pregnancy.

Figure 2 depicts the decision function of this classifier, in the form of an Ordered Binary Decision Diagram (OBDD). Given some test results, we make a corresponding decision on pregnancy by simply navigating the OBDD. We start at the root, which is labeled with the Urine (U) test. Depending on the outcome of this test, we follow the edge labeled positive, or the edge labeled negative. We repeat for the test labeled at the next node. Eventually, we reach a leaf node labeled "yes" or "no," which provides the resulting classification.

The decisions rendered by this OBDD are guaranteed to match those obtained from the naive Bayes classifier. We have thus converted a probabilistic classifier into an equivalent classifier that is symbolic and tractable. We will later see how this facilitates the efficient generation of explanations.

We will later discuss compiling Bayesian network classifiers into ODDs, after formally treating classifiers and ODDs.

2.1 Bayesian Network Classifiers

A *Bayesian network classifier* is a Bayesian network containing a special set of variables: a single *class* variable C and n *feature* variables $\mathbf{X} = \{X_1, \ldots, X_n\}$. The class C is usually a root in the network and the features $\mathbf{X}$ are usually leaves. In this paper, we assume that the class variable is binary, with two values c and $\bar{c}$ that correspond to positive and negative classes, respectively (i.e., "yes" and "no" decisions). An instantiation of variables $\mathbf{X}$ is denoted $\mathbf{x}$ and called an *instance*. A Bayesian network classifier specifying probability distribution $Pr(.)$ will classify an instance $\mathbf{x}$ positively iff $Pr(c \mid \mathbf{x}) \geq T$, where T is called the classification *threshold*.

Definition 1 (Decision Function) *Suppose that we have a Bayesian network classifier with features $\mathbf{X}$, class variable C and a threshold T. Let $f(\mathbf{X})$ be a function that maps instances $\mathbf{x}$ into $\{0, 1\}$. We say that $f(\mathbf{X})$ is the classifier's decision function iff*

$$f(\mathbf{x}) = \begin{cases} 1 & \text{if } Pr(c \mid \mathbf{x}) \geq T \\ 0 & \text{otherwise.} \end{cases}$$

Instance $\mathbf{x}$ is positive if $f(\mathbf{x}) = 1$ and negative if $f(\mathbf{x}) = 0$.

The *naive Bayes classifier* is a special type of a Bayesian network classifier, where edges extend from the class to features (no other nodes or edges). Figure 1 depicted a naive Bayes classifier. A *latent-tree classifier* is a tree-structured Bayesian network, whose root is the class variable and whose leaves are the features.

2.2 Monotone Classifiers

The class of *monotone* classifiers is relevant to our discussion, particularly when relating the two types of explanations we shall propose. We will define these classifiers next, while assuming binary features to simplify the treatment. Intuitively, a monotone classifier satisfies the following. A positive instance remains positive if we flip some of its features from 0 to 1. Moreover, a negative instance remains negative if we flip some of its features from 1 to 0.

More formally, consider two instances $\mathbf{x}^\star$ and $\mathbf{x}$. We write $\mathbf{x}^\star \subseteq^1 \mathbf{x}$ to mean: the features set to 1 in $\mathbf{x}^\star$ is a subset of those set to 1 in $\mathbf{x}$. Monotone classifiers are then characterized by the following property of their decision functions, which is well-known in the literature on Boolean functions.

Definition 2 *A decision function $f(\mathbf{X})$ is monotone iff*

$$\mathbf{x}^\star \subseteq^1 \mathbf{x} \quad \text{only if} \quad f(\mathbf{x}^\star) \leq f(\mathbf{x}).$$

One way to read the above formal definition is as follows. If the positive features in instance $\mathbf{x}$ contain those in instance $\mathbf{x}^\star$, then instance $\mathbf{x}$ must be positive if instance $\mathbf{x}^\star$ is positive.

It is generally difficult to decide whether a Bayesian network classifier is monotone; see, e.g., [van der Gaag *et al.*, 2004]. However, if the decision function of the classifier is an OBDD, then monotonicity can be decided in time quadratic in the OBDD size [Horiyama and Ibaraki, 2002].

[3] Available at `https://arxiv.org/abs/1805.03364`.

2.3 Ordered Decision Diagrams

An Ordered Binary Decision Diagram (OBDD) is based on an ordered set of binary variables $\mathbf{X} = X_1, \ldots, X_n$. It is a rooted, directed acyclic graph, with two sinks called the 1-sink and 0-sink. Every node (except the sinks) in the OBDD is labeled with a variable X_i with two outgoing edges, one labeled 1 and the other labeled 0. If there is an edge from a node labeled X_i to a node labeled X_j, then $i < j$. An OBDD is defined over binary variables, but can be extended to discrete variables with arbitrary values. This is called an ODD: a node labeled with variable X_i has one outgoing edge for each value of variable X_i. Hence, an OBDD/ODD can be viewed as representing a function $f(\mathbf{X})$ that maps instances $\mathbf{x}$ into $\{0, 1\}$. Figure 2 depicted an OBDD. Note: in this paper, we use positive/yes/1 and negative/no/0 interchangeably.

An OBDD is a *tractable* representation of a function $f(\mathbf{X})$ as it can be used to efficiently answer many queries about the function. For example, one can in linear time count the number of positive instances $\mathbf{x}$ (i.e., $f(\mathbf{x}) = 1$), called the *models* of f. One can also conjoin, disjoin and complement OBDDs efficiently. This tractability, which carries over to ODDs, will be critical for efficiently generating explanations. For more on OBDDs, see [Meinel and Theobald, 1998; Wegener, 2000].

2.4 Compiling Decision Functions

[Chan and Darwiche, 2003] proposed an algorithm for compiling a naive Bayes classifier into an ODD, while guaranteeing an upper bound on the time of compilation and the size of the resulting ODD. In particular, for a classifier with n features, the compiled ODD has a number of nodes that is bounded by $O(b^{\frac{n}{2}})$ and can be obtained in time $O(nb^{\frac{n}{2}})$. Here, b is the maximum number of values that a variable may have. The actual time and space complexity can be much less, depending on the classifier's parameters and variable order used for the ODD (as observed experimentally).

The algorithm is based on the following insights. Let $\mathbf{X}$ be all features. Observing features $\mathbf{Y} \subset \mathbf{X}$ leads to another naive Bayes classifier, with features $\mathbf{X} \setminus \mathbf{Y}$ and an adjusted class prior. Consider now a decision tree over features $\mathbf{X}$ and a node in the tree that was reached by a partial instantiation $\mathbf{y}$. We annotate this node with the corresponding naive Bayes classifier $N_{\mathbf{y}}$ found by observing $\mathbf{y}$, and then merge nodes with equivalent classifiers—those having equivalent decision functions—as described by [Chan and Darwiche, 2003]. Implementing this idea carefully leads to an ordered decision diagram (ODD) with the corresponding bounds.[4]

Algorithm 1 is a simpler variation on the algorithm of [Chan and Darwiche, 2003]; it has the same complexity bounds, but may be less efficient in practice. It uses procedure `expand-then-merge`$(., D, X)$, which expands the partial decision graph D by a feature X, then merges nodes that correspond to equivalent classifiers.

Using this procedure, we propose Algorithm 2 for compiling a latent-tree classifier into an ODD. Here's the key insight. Let R be a node in a latent-tree classifier where all features outside R have been observed, and let C be a child of R. Observing all features under C leads to a new latent-tree classifier without the subtree rooted at C and an adjusted class prior. Algorithm 2 uses this observation by iteratively choosing a node C and then shrinking the classifier size by instantiating the features under C, allowing us to compile an ODD in a fashion similar to [Chan and Darwiche, 2003]. The specific choice of internal nodes C by Algorithm 2 leads to the following complexity.

[4] [Chan and Darwiche, 2003] uses a sophisticated, but conceptually simple, technique for identifying equivalent classifiers.

Algorithm 1 `compile-naive-bayes`(N)

input: A naive Bayes classifier N

output: An ODD for the decision function of N

main:

```
1: D ← empty decision graph
2: for each feature X of classifier N do
3:    D ← expand-then-merge(N, D, X)
4: return ODD D
```

Algorithm 2 `compile-latent-tree`(N)

input: A latent-tree classifier N

output: An ODD for the decision function of N

main:

```
1: D ← empty decision graph
2: R ← root of tree N
3: while R has unprocessed children do
4:    if R has a single internal and unprocessed child C
      then
5:       R ← C
6:    else
7:       C ← child of R with smallest number of leaves
8:       for each leaf X under C do
9:          D ← expand-then-merge(N, D, X)
10:      mark C as processed
11: return ODD D
```

Theorem 1 *Given a latent-tree classifier N with n variables, each with at most b values, the ODD computed by Algorithm 2 has size $O(b^{\frac{3n}{4}})$ and can be obtained in time $O(nb^{\frac{3n}{4}})$.*

If one makes further assumptions about the structure of the latent tree (e.g., if the root has k children, and each child of the root has $O(\frac{n}{k})$ features), then one obtains the size bound of $O(b^{\frac{n}{2}})$ and time bound of $O(nb^{\frac{n}{2}})$ for naive Bayes classifiers. We do not expect a significantly better upper bound on the time complexity due to the following result.

Theorem 2 *Given a naive Bayes classifier N, compiling an ODD representing its decision function is NP-hard.*

3 Minimum Cardinality Explanations

We now consider the first type of explanations for why a classifier makes a certain decision. These are called *minimum-cardinality explanations* or MC-explanations. We will first assume that the features are binary and then generalize later.

Consider two instances $\mathbf{x}^\star$ and $\mathbf{x}$. As we did earlier, we write $\mathbf{x}^\star \subseteq^1 \mathbf{x}$ to mean: the features set to 1 in $\mathbf{x}^\star$ are a subset of those set to 1 in $\mathbf{x}$. We define $\mathbf{x}^\star \subseteq^0 \mathbf{x}$ analogously. Moreover, we write $\mathbf{x} \leq^1 \mathbf{x}^\star$ to mean: the count of 1-features in $\mathbf{x}$ is no greater than their count in $\mathbf{x}^\star$. We define $\mathbf{x} \leq^0 \mathbf{x}^\star$ analogously.

Definition 3 (MC-Explanation) *Let $f(\mathbf{X})$ be a given decision function. An MC-explanation of a positive instance $\mathbf{x}$ is another positive instance $\mathbf{x}^\star$ such that $\mathbf{x}^\star \subseteq^1 \mathbf{x}$ and there is no other positive instance $\mathbf{x}' \subseteq^1 \mathbf{x}$ where $\mathbf{x}' <^1 \mathbf{x}^\star$. An MC-explanation of a negative instance $\mathbf{x}$ is another negative instance $\mathbf{x}^\star$ such that $\mathbf{x}^\star \subseteq^0 \mathbf{x}$ and there is no other negative instance $\mathbf{x}' \subseteq^0 \mathbf{x}$ where $\mathbf{x}' <^0 \mathbf{x}^\star$.*

Intuitively, an MC-explanation of a positive decision $f(\mathbf{x}) = 1$ answers the question: which positive features of instance $\mathbf{x}$ are responsible for this decision? Similarly for the MC-explanation of a negative decision $f(\mathbf{x}) = 0$: which negative features of instance $\mathbf{x}$ are responsible for this decision? MC-explanations are not necessarily unique as we shall see later. However, MC-explanations of positive decisions must all have the same number of 1-features, and those for negative decisions must all have the same number of 0-features.

MC-explanations are perhaps best illustrated using a monotone classifier. As a running example, consider a (monotone) classifier for deciding whether a student will be admitted to a university. The class variable is admit (A) and the features of an applicant are:

- work-experience (W): has prior work experience.
- first-time-applicant (F): did not apply before.
- entrance-exam (E): passed the entrance exam.
- gpa (G): has met the university's expected GPA.

All variables are either positive (+) or negative (-).

Consider a naive Bayes classifier with the following false positive and false negative rates:

feature	f_p	f_n
W	0.10	0.04
F	0.20	0.30
E	0.15	0.60
G	0.11	0.03

To completely specify the naive Bayes classifier, we also need the prior probability of admission, which we assume to be $Pr(A{=}+) = 0.30$. Moreover, we use a decision threshold of 0.50, admitting an applicant $\mathbf{x}$ if $Pr(A{=}+ \mid \mathbf{x}) \geq .50$. Note that with the above false positive and false negative rates, a positively observed feature will increase the probability of a positive classification, while a negatively observed feature will increase the probability of a negative classification (hence, the classifier is monotone).

Table 1 depicts the decision function f for this naive Bayes classifier, with MC-explanations for all 16 instances.

Consider, for example, a student (+ + + +) who was admitted by this decision function. There is a single MC-explanation for this decision, (+ - - +), with cardinality 2. According to this explanation, work experience and a good GPA were the reasons for admission. That is, the student would still have been admitted even if they have applied before and did not pass the entrance exam.

W	F	E	G	$Pr(A{=}+\|\mathbf{x})$	$f(\mathbf{x})$	MC-explanations
-	-	-	-	0.0002	-	(- - + +) (- + - +) (- + + -) (+ - + -) (+ + - -)
-	-	-	+	0.0426	-	(- - + +) (- + - +)
-	-	+	-	0.0006	-	(- - + +) (- + + -) (+ - + -)
-	-	+	+	0.1438	-	(- - + +)
-	+	-	-	0.0016	-	(- + - +) (- + + -) (+ + - -)
-	+	-	+	0.2933	-	(- + - +)
-	+	+	-	0.0060	-	(- + + -)
-	+	+	+	0.6105	+	(- + + +)
+	-	-	-	0.0354	-	(+ + - -) (+ - + -)
+	-	-	+	0.9057	+	(+ - - +)
+	-	+	-	0.1218	-	(+ - + -)
+	-	+	+	0.9732	+	(+ - - +)
+	+	-	-	0.2552	-	(+ + - -)
+	+	-	+	0.9890	+	(+ - - +)
+	+	+	-	0.5642	+	(+ + + -)
+	+	+	+	0.9971	+	(+ - - +)

Table 1: A decision function with MC-explanations.

For another example, consider a student (- - - +) who was rejected. There are two MC-explanations for this decision. The first, (- - + +), says that the student would not have been admitted, even if they passed the entrance exam. The second explanation, (- + - +), says that the student would not have been admitted, even if they were a first-time applicant.

Finally, we remark that while MC-explanations are more intuitive for monotone classifiers, they also apply to classifiers that are not monotone, as we shall see in Section 3.2.

3.1 Computing MC-Explanations

We will now present an efficient algorithm for computing the MC-explanations of a decision, assuming that the decision function has a specific form. Our treatment assumes that the decision function is represented as an OBDD, but it actually applies to a broader class of representations which includes OBDDs as a special case. More on this later.

Our algorithm uses a key operation on decision functions.

Definition 4 (Cardinality Minimization) *For $i \in \{0, 1\}$, the i-minimization of decision function $f(\mathbf{X})$ is another decision function $f^i(\mathbf{X})$ defined as follows: $f^i(\mathbf{x}) = 1$ iff (a) $f(\mathbf{x}) = 1$ and (b) $\mathbf{x} \leq^i \mathbf{x}^\star$ for every $f(\mathbf{x}^\star) = 1$.*

The 1-minimization of decision function f renders positive decisions only on the positive instances of f having a minimal number of 1-features. Similarly, the 0-minimization of decision function f renders positive decisions only on the positive instances of f having a minimal number of 0-features. Cardinality minimization was discussed and employed for other purposes in [Darwiche, 2001; Choi *et al.*, 2013].

Algorithm 3 computes the MC-explanations of a decision $f(\mathbf{x})$. The set of computed explanations is encoded by another decision function $g(\mathbf{X})$. In particular, $g(\mathbf{x}^\star) = 1$ iff $\mathbf{x}^\star$ is an MC-explanation of decision $f(\mathbf{x})$.

Suppose we want to compute the MC-explanations of a positive decision $f(\mathbf{x}) = 1$. The algorithm will first find the portion α of instance $\mathbf{x}$ with variables set to 0. It will then

Algorithm 3 `find-mc-explanation`$(f(\mathbf{X}), \mathbf{x})$

input: An OBDD $f(\mathbf{X})$ and instance $\mathbf{x}$.

output: An OBDD $g(\mathbf{X})$ where $g(\mathbf{x}^\star) = 1$ iff $\mathbf{x}^\star$ is an MC-explanation of decision $f(\mathbf{x})$.

main:

1: $i \leftarrow f(\mathbf{x})$
2: $\alpha \leftarrow$ the subset of $\mathbf{x}$ with variables set to $1 - i$
3: complement function f if $i = 0$
4: **return** i-minimize(conjoin(f, α))

conjoin[5] f with α and 1-minimize the result. The obtained decision function encodes the MC-explanations in this case.

An OBDD can be complemented and conjoined with a variable instantiation in linear time. It can also be minimized in linear time. This leads to the following complexity for generating MC-explanations based on OBDDs.

Theorem 3 *When the decision function $f(\mathbf{X})$ is represented as an OBDD, the time and space complexity of Algorithm 3 is linear in the size of f, while guaranteeing that the output function $g(\mathbf{X})$ is also an OBDD.*

Given OBDD properties, one can count MC-explanations in linear time, and enumerate each in linear time.

3.2 Case Study: Votes Classifier

We now consider the Congressional Voting Records (votes) from the UCI machine learning repository [Bache and Lichman, 2013]. This dataset consists of 16 key votes by Congressmen of the U.S. House of Representatives. The class label is the party of the Congressman (positive if Republican and negative if Democrat). A naive Bayes classifier trained on this dataset obtains 91.0% accuracy. We compiled this classifier into an OBDD, which has a size of 630 nodes.

The following Congressman from the dataset voted on all 16 issues and was classified correctly as a Republican:

(0 1 0 1 1 1 0 0 0 0 0 0 1 1 0 1)

This decision has five MC-explanations of cardinality 3, e.g.:

(0 0 0 1 1 0 0 0 0 0 0 0 0 1 0 0)

The MC-explanation tells us that this Congressmen could have reversed four of their yes-votes, and the classifier would still predict that this Congressman was a Republican.

For a problem of this size, we can enumerate all instances of the classifier. We computed the MC-explanations for each of the $32,256$ positive instances, out of a possible number of $2^{16} = 65,536$ instances. Among these MC-explanations, the one that appeared the most frequently was the MC-explanation from the above example. This explanation corresponded to yes-votes on three issues: physician-fee-freeze, el-salvador-aid, and crime. Further examination of the dataset revealed that these issues were the three with the fewest Republican no-votes.

[5]Conjoining f with α leads to a function h such that $h(\mathbf{x}) = 1$ iff $f(\mathbf{x}) = 1$ and $\mathbf{x}$ is compatible with α.

W	F	E	G	$Pr(A{=}+\|\mathbf{x})$	$f(\mathbf{x})$	PI-explanations
-	-	-	-	0.0002	-	$(\bar{w}\bar{f})$ $(\bar{w}\bar{e})$ $(\bar{w}\bar{g})$ $(\bar{f}\bar{g})$ $(\bar{e}\bar{g})$
-	-	-	+	0.0426	-	$(\bar{w}\bar{f})$ $(\bar{w}\bar{e})$
-	-	+	-	0.0006	-	$(\bar{w}\bar{f})$ $(\bar{w}\bar{g})$ $(\bar{f}\bar{g})$
-	-	+	+	0.1438	-	$(\bar{w}\bar{f})$
-	+	-	-	0.0016	-	$(\bar{w}\bar{e})$ $(\bar{w}\bar{g})$ $(\bar{e}\bar{g})$
-	+	-	+	0.2933	-	$(\bar{w}\bar{e})$
-	+	+	-	0.0060	-	$(\bar{w}\bar{g})$
-	+	+	+	0.6105	+	(feg)
+	-	-	-	0.0354	-	$(\bar{f}\bar{g})$ $(\bar{e}\bar{g})$
+	-	-	+	0.9057	+	(wg)
+	-	+	-	0.1218	-	$(\bar{f}\bar{g})$
+	-	+	+	0.9732	+	(wg)
+	+	-	-	0.2552	-	$(\bar{e}\bar{g})$
+	+	-	+	0.9890	+	(wg)
+	+	+	-	0.5642	+	(wfe)
+	+	+	+	0.9971	+	(wg) (wfe) (feg)

Table 2: A decision function with PI-explanations.

4 Prime Implicant Explanations

We now consider the second type of explanations, called *prime-implicant explanations* or PI-explanations for short.

Let $\mathbf{y}$ and $\mathbf{z}$ be instantiations of some features and call them *partial instances*. We will write $\mathbf{y} \supseteq \mathbf{z}$ to mean that $\mathbf{y}$ extends $\mathbf{z}$, that is, it includes $\mathbf{z}$ but may set some additional features.

Definition 5 (PI-Explanation) *Let $f(\mathbf{X})$ be a given decision function. A PI-explanation of a decision $f(\mathbf{x})$ is a partial instance $\mathbf{z}$ such that*

(a) $\mathbf{z} \subseteq \mathbf{x}$,

(b) $f(\mathbf{x}) = f(\mathbf{x}^\star)$ *for every* $\mathbf{x}^\star \supseteq \mathbf{z}$, *and*

(c) no other partial instance $\mathbf{y} \subset \mathbf{z}$ *satisfies (a) and (b).*

Intuitively, a PI-explanation of decision $f(\mathbf{x})$ is a minimal subset $\mathbf{z}$ of instance $\mathbf{x}$ that makes features outside $\mathbf{z}$ irrelevant to the decision. That is, we can toggle any feature that does not appear in $\mathbf{z}$ while maintaining the current decision. The number of features appearing in a PI-explanation will be called the *length* of the explanation. As we shall see later, PI-explanations of the same decision may have different lengths.

Table 2 depicts the decision function f for the admissions classifier, with PI-explanations for all 16 instances. We write (wg) for $W{=}+, G{=}+$ and $(\bar{e}\bar{g})$ for $E{=}-, G{=}-$.

Consider a student (+ + - -) who was *not* admitted by this decision function. There is a single PI-explanation $(\bar{e}\bar{g})$ for this decision. According to this explanation, it is sufficient to have a poor entrance exam and a poor GPA to be rejected—it does not matter whether they have work experience or if they are a first-time applicant. That is, we can set these features to any value, and the applicant would still be rejected.

Consider now a student (+ + + +) who was admitted. There are three PI-explanations for this decision, (wg) (wfe) (feg), with different lengths. These explanations can be visualized as (+ * * +), (+ + + *) and (* + + +). This is in contrast to the single MC-explanation (+ - - +) obtained previously.

4.1 Computing Prime Implicant Explanations

Algorithms exist for converting an OBDD for function f into an ODD that encodes the prime implicants of f; see, e.g.,

[Coudert and Madre, 1993; Coudert *et al.*, 1993; Minato, 1993]. These algorithms recurse on the structure of the input OBDD, computing prime implicants of sub-OBDDs. As we are interested in explaining a specific instance $\mathbf{x}$, we only need the prime implicants compatible with $\mathbf{x}$ (a function may have exponentially many prime implicants, but those compatible with an instance may be small). In the full version of this paper [Shih *et al.*, 2018], we exploit this observation by providing a more efficient algorithm which computes the PI-explanations of a given positive instance $\mathbf{x}$, by looking for prime implicants only in the relevant sub-OBDDs. Empirically, such an algorithm can be twice as fast, and can generate ODDs that are an order-of-magintude smaller, compared to finding all prime implicants.

4.2 Case Study: Votes Classifier

Consider again the voting record of the Republican Congressman that we considered earlier in Section 3.2:

$$(0\,1\,0\,1\,1\,1\,0\,0\,0\,0\,0\,0\,1\,1\,0\,1)$$

There are 30 PI-explanations of this decision. There are 2 shortest explanations of 9 features:

$$\begin{array}{lll} (& 0\,1\,1 \quad 0\,0\,0 & 1\,1\,0 \;\;) \\ (& 0\,1\,1\,1 \quad 0\,0 & 1\,1\,0 \;\;) \end{array}$$

The first corresponds to yes votes on:

physician-fee-freeze, el-salvador-aid,
superfund-right-to-sue, crime,

and no votes on

adoption-of-the-budget-resolution, anti-satellite-test-ban,
aid-to-nicaraguan-contras, mx-missile, duty-free-exports.

These 9 votes necessitate the classification of a Republican; no other vote changes this decision.

5 More On Monotone Classifiers

We now discuss a specific relationship between MC and PI explanations for monotone classifiers.

An MC-explanation sets all features, while a PI-explanation sets only a subset of the features. For a positive instance, we will say that MC-explanation $\mathbf{x}$ and PI-explanation $\mathbf{z}$ *match* iff $\mathbf{x}$ can be obtained from $\mathbf{z}$ by setting all missing features negatively. For a negative instance, MC-explanation $\mathbf{x}$ and PI-explanation $\mathbf{z}$ match iff $\mathbf{x}$ can be obtained from $\mathbf{z}$ by setting all missing features positively.

Theorem 4 *For a decision $f(\mathbf{x})$ of a monotone decision function f: (1) Each MC-explanation matches some shortest PI-explanation. (2) Each shortest PI-explanation matches some MC-explanation.*

Hence, for monotone decision functions, MC-explanations coincide with shortest PI-explanations.

The admissions classifier we considered earlier is monotone, which can be verified by inspecting its decision function (in contrast, the votes classifier is not monotone). Here, all MC-explanations matched PI-explanations. For example, the MC-explanation (+ - - +) for instance (+ + - +) matches the PI-explanation (wg). However, the PI-explanation (wfe) for instance (+ + + +) does not match the single MC-explanation (+ - - +). One can verify though, by examining Tables 1 and 2, that shortest PI-explanations coincide with MC-explanations.

MC-explanations are no longer than PI-explanations and their count is no larger than the count of PI-explanations. Moreover, MC-explanations can be computed in linear time, given that the decision function is represented as an OBDD. This is not guaranteed for PI-explanations.

6 Related Work

There has been significant interest recently in providing explanations for classifiers; see, e.g., [Ribeiro *et al.*, 2016b; Elenberg *et al.*, 2017; Lundberg and Lee, 2017; Ribeiro *et al.*, 2016a; 2018]. In particular, *model-agnostic* explainers were sought [Ribeiro *et al.*, 2016b], which can explain the behavior of (most) any classifier, by treating it as a *black box*. Take for example, LIME, which *locally* explains the classification of a given instance. Roughly, LIME samples new instances that are "close" to a given instance, and then learns a simpler, interpretable model from the sampled data. For example, suppose a classifier rejects a loan to an applicant; one could learn a decision tree for other instances similar to the applicant, to understand why the original decision was made.

More related to our work is the notion of an "anchor" introduced in [Ribeiro *et al.*, 2016a; 2018]. An anchor for an instance is a subset of the instance that is highly likely to be classified with the same label, no matter how the missing features are filled in (according to some distribution). An anchor can be viewed as a probabilistic extension of a PI-explanation. Anchors can also be understood using the Same-Decision Probability (SDP) [Choi *et al.*, 2012; Chen *et al.*, 2014; Choi *et al.*, 2017], proposed in [Darwiche and Choi, 2010]. In this context, the SDP asks, "Given that I have already observed $\mathbf{x}$, what is the probability that I will make the same classification if I observe the remaining features?" In this case, we expect an anchor $\mathbf{x}$ to have a high SDP, but a PI-explanation $\mathbf{x}$ will always have an SDP of 1.0.

7 Conclusion

We proposed an algorithm for compiling latent-tree Bayesian network classifiers into decision functions in the form of ODDs. We also proposed two approaches for explaining the decision that a Bayesian network classifier makes on a given instance, which apply more generally to any decision function in symbolic form. One approach is based on MC-explanations, which minimize the number of positive features in an instance, while maintaining its classification. The other approach is based on PI-explanations, which identify a smallest set of features in an instance that renders the remaining features irrelevant to a classification. We proposed algorithms for computing these explanations when the decision function has a symbolic and tractable form. We also discussed monotone classifiers and showed that MC-explanations and PI-explanations coincide for this class of classifiers.

Acknowledgments

This work has been partially supported by NSF grant #IIS-1514253, ONR grant #N00014-15-1-2339 and DARPA XAI grant #N66001-17-2-4032.

References

[Bache and Lichman, 2013] K. Bache and M. Lichman. UCI machine learning repository, 2013.

[Bryant, 1986] R. E. Bryant. Graph-based algorithms for Boolean function manipulation. *IEEE Transactions on Computers*, C-35:677–691, 1986.

[Chan and Darwiche, 2003] Hei Chan and Adnan Darwiche. Reasoning about Bayesian network classifiers. In *Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 107–115, 2003.

[Chen *et al.*, 2014] Suming Chen, Arthur Choi, and Adnan Darwiche. Algorithms and applications for the same-decision probability. *Journal of Artificial Intelligence Research*, 49:601–633, 2014.

[Choi *et al.*, 2012] Arthur Choi, Yexiang Xue, and Adnan Darwiche. Same-decision probability: A confidence measure for threshold-based decisions. *International Journal of Approximate Reasoning (IJAR)*, 53(9):1415–1428, 2012.

[Choi *et al.*, 2013] Arthur Choi, Doga Kisa, and Adnan Darwiche. Compiling probabilistic graphical models using sentential decision diagrams. In *Proceedings of the 12th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU)*, pages 121–132, 2013.

[Choi *et al.*, 2017] YooJung Choi, Adnan Darwiche, and Guy Van den Broeck. Optimal feature selection for decision robustness in Bayesian networks. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)*, August 2017.

[Coudert and Madre, 1993] Olivier Coudert and Jean Christophe Madre. Fault tree analysis: 10^{20} prime implicants and beyond. In *Proc. of the Annual Reliability and Maintainability Symposium*, 1993.

[Coudert *et al.*, 1993] Olivier Coudert, Jean Christophe Madre, Henri Fraisse, and Herve Touati. Implicit prime cover computation: An overview. In *Proceedings of the 4th SASIMI Workshop*, 1993.

[Darwiche and Choi, 2010] Adnan Darwiche and Arthur Choi. Same-decision probability: A confidence measure for threshold-based decisions under noisy sensors. In *Proceedings of the Fifth European Workshop on Probabilistic Graphical Models (PGM)*, pages 113–120, 2010.

[Darwiche, 2001] Adnan Darwiche. Decomposable negation normal form. *Journal of the ACM*, 48(4):608–647, 2001.

[Elenberg *et al.*, 2017] Ethan R. Elenberg, Alexandros G. Dimakis, Moran Feldman, and Amin Karbasi. Streaming weak submodularity: Interpreting neural networks on the fly. In *Advances in Neural Information Processing Systems 30 (NIPS)*, pages 4047–4057, 2017.

[Horiyama and Ibaraki, 2002] Takashi Horiyama and Toshihide Ibaraki. Ordered binary decision diagrams as knowledge-bases. *Artificial Intelligence (AIJ)*, 136(2):189–213, 2002.

[Lundberg and Lee, 2017] Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems 30 (NIPS)*, pages 4768–4777, 2017.

[Meinel and Theobald, 1998] Christoph Meinel and Thorsten Theobald. *Algorithms and Data Structures in VLSI Design: OBDD — Foundations and Applications*. Springer, 1998.

[Minato, 1993] Shin-ichi Minato. Fast generation of prime-irredundant covers from binary decision diagrams. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 76(6):967–973, 1993.

[Ribeiro *et al.*, 2016a] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Nothing else matters: Model-agnostic explanations by identifying prediction invariance. In *NIPS Workshop on Interpretable Machine Learning in Complex Systems*, 2016.

[Ribeiro *et al.*, 2016b] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should i trust you?": Explaining the predictions of any classifier. In *Knowledge Discovery and Data Mining (KDD)*, 2016.

[Ribeiro *et al.*, 2018] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Anchors: High-precision model-agnostic explanations. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI)*, 2018.

[Shih *et al.*, 2018] Andy Shih, Arthur Choi, and Adnan Darwiche. A symbolic approach to explaining Bayesian network classifiers. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI)*, 2018.

[van der Gaag *et al.*, 2004] Linda C. van der Gaag, Hans L. Bodlaender, and A. J. Feelders. Monotonicity in Bayesian networks. In *Proceedings of the 20th Conference in Uncertainty in Artificial Intelligence (UAI)*, pages 569–576, 2004.

[Wegener, 2000] Ingo Wegener. *Branching Programs and Binary Decision Diagrams*. SIAM, 2000.

Towards a Taxonomy for Interpretable and Interactive Machine Learning

Elio Ventocilla[1], **Tove Helldin**[1], **Maria Riveiro**[1], **Juhee Bae**[1], **Veselka Boeva**[2], **Göran Falkman**[1], **Niklas Lavesson**[3]
[1] University of Skövde, School of Informatics
[2] Blekinge Institute of Technology, Department of Computer Science and Engineering
[3] Jönköping University, School of Engineering
{elio.ventocilla, tove.helldin, maria.riveiro, juhee.bae, goran.falkman}@his.se
veselka.boeva@bth.se
niklas.lavesson@ju.se

Abstract

We propose a taxonomy for classifying and describing papers which contribute to making Machine Learning (ML) techniques interactive and interpretable for users. The taxonomy is composed of six elements – Dataset, Optimizer, Model, Predictions, Evaluator and Goodness – where each can be made available for user interpretation and interaction. We give definitions to the terms interpretable and interactive in the context of user-oriented Machine Learning, describe the role of each of the elements in the taxonomy, and describe papers as seen through the lens of the proposed taxonomy.

1 Introduction

Due to the availability of large datasets, we enter a new era of *augmented intelligence*, where machines support humans to increase their cognitive capabilities. Indeed, when problems are complex and ill-defined, user-ML cooperation is needed. This approach to problem-solving is appealing for many reasons, for instance, to integrate valuable expert knowledge that may be hard to encode directly into computational models, to help resolve existing uncertainties as a result of error that may arise from automatic ML, or to build trust by making humans involved in the modeling or learning ML processes [Boukhelifa *et al.*, 2018]. A human and a machine collaborate to achieve a task, whether this is to classify objects, to find interesting data projections or patterns, or to design creative artworks [Boukhelifa *et al.*, 2018]. To date, several researchers have recently started to work at the intersection of human-computer interaction (HCI) and ML where the interaction with humans is seen as a central part of developing ML-systems.

The rapid increase of works related to interpretable and interactive machine learning (iiML) calls for an overview of this interdisciplinary subject, in order to structure the current literature and to develop a research agenda. One recent example of this type of study is presented in [Abdul *et al.*, 2018], focusing on providing an HCI agenda for explainable, accountable and intelligible systems. Another recent study [Lipton, 2016] focuses on interpretability in supervised learning and conducts a critical analysis of the literature to improve the specification of the task of interpretation. Interactive learning is explicitly put as out of scope in the paper.

In this paper, we unify the interactive and interpretable perspectives as we view them linked and in many cases interdependent. We tackle the challenge of classifying these works from an ML-component perspective, where we look at the components themselves and try to organize the studies found in the literature regarding the components that are being made interpretable or interactive, and how.

Thus, the main contribution of this paper is to present a taxonomy for categorizing the literature in the area of iiML. This taxonomy can be used to (1) provide a structured overview of the research work in the area of iiML, (2) identify research trends and opportunities for research in the future and (3) suggest a standard terminology for iiML.

The paper describes, first, relevant background and terminology in the context of iiML (see section 2). The method followed for outlining the taxonomy is summarized in section 3. The taxonomy and its components are described thereafter in section 4. We discuss implications in section 5 and conclude the paper with some conclusions and final remarks in section 6.

2 Background concepts

The use of ML-based support systems has shown great potential in multiple areas such as education, healthcare, manufacturing, retail, etc. To fully exploit the benefits of such systems in our daily activities, we need to make these technologies more accessible to all. Nevertheless, many conventional applications of ML are mostly agnostic to the fact that their inputs and outputs are directed at humans [Amershi *et al.*, 2013]. To resolve this, researchers from the areas of HCI and AI now collaborate, trying to make these systems more transparent and understandable, providing explanations, visualizing the inner workings of these complex ML systems or supporting human interaction with the core elements of ML.

Several concepts and terms have arisen in this quest, for instance, *interpretability*, *interactivity*, *understandability*, *explainability*, *comprehensibility*, *intelligibility* and *transparency*. This section gives a very brief summary of some definitions found in the literature for these terms, in order to

provide an introduction of the rich terminology used by researchers within the iiML area.

Interpretation in relation to ML techniques is defined by Chuang *et al.* [2012] as the "facility with which an analyst makes inferences about the underlying data". The term *transparent*, on the other hand, is related to ML techniques that, (1) produce models that a typical real-world user can read and understand; (2) use algorithms that a typical real-world user can understand, and (3) allow a real-world user to adapt models to new domains [Chiticariu *et al.*, 2015].

Comprehensibility, *understandability* and *interpretability* are regarded as synonyms in [Piltaver *et al.*, 2016]. Comprehensibility is defined by Zhou [2005] as a property of ML algorithms that "produce patterns understandable to human beings". In the same paper the author refers to a postulate made by Michalski [1983], which states that the resulting elements of a computer induction "should be comprehensible as single 'chunks' of information, directly interpretable in natural language". Moreover, the term *explainable* is found to be used in conjunction with the terms intelligibility [Abdul *et al.*, 2018], understandable [Stumpf *et al.*, 2009] and transparent [Lim *et al.*, 2009].

Most often the terms above are used to describe ML components that readily lend themselves, or are presented in a way, which is understandable to humans. For a closer discussion of the terms, please refer to [Bibal and Frénay, 2016].

The term *interactive* is, in this context, referred by Fails and Olsen [2003] as a property of models which allow users to "train, classify/view and correct the classifications". Holzinger *et al.* [2016] expand the interaction boundaries from users to agents, where agents can also be other systems. Fiebrink *et al.* [2011] highlight the importance of model evaluation for effective user interaction in model improvement. A term related to interactive ML often found in the literature is that of human-in-the-loop (e.g. [Holzinger *et al.*, 2016; Bohanec *et al.*, 2017; Lee *et al.*, 2017]), and is used to describe a role played by users in improving the "goodness" of a model by giving feedback to the ML algorithm, and/or "steering" its computation.

In our work, we use interpretable and interactive ML as an overall term that comprises all of the above concepts, where focus is put on placing the human user in the center of the ML processes. As we see it, interpretability is enabled through an ML system's explainability, transparency, intelligibility and understandability, and is often realized through the user interacting with the ML system, enabling the user to obtain a better understanding of the system's inner workings as well as to improve its output.

3 Method

The taxonomy has been developed through the following process: (1) *keyword and query definition*, i.e., establishing search keywords with which papers were queried in the databases; (2) *paper ranking*, i.e., sorting and choosing papers based on a given criteria; (3) *paper survey*, i.e., individually surveying the contents of the papers based on their relation to ML interaction and interpretation; (4) and *discussion*, i.e., coming to an agreement between all authors about common elements found in step (3). The last two steps, *paper survey* and *discussion*, were iterated as often as needed for all authors to come to a consensus on the common elements found and their relations.

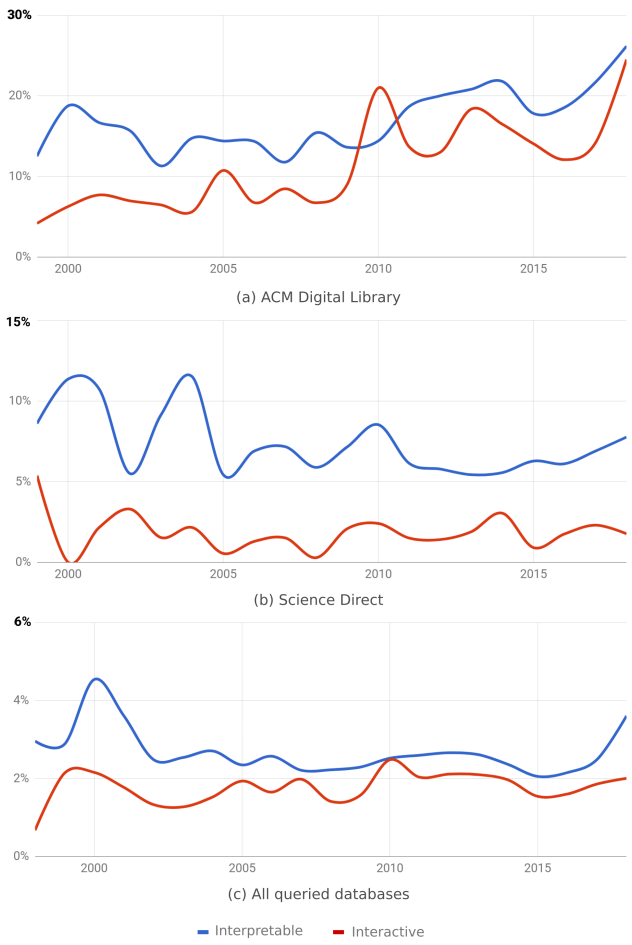


Figure 1: Trend over the past 20 years of ML journal and conference publications matching our queries on interpretable and interactive ML. These represent results from (a) ACM Digital Library, (b) Science Direct, and (c) all queried databases, i.e., ACM Digital Library, Science Direct, IEEE Xplore, Scopus, Web of Science, NIPS and MLR. The y-axis depicts percentage, that is, the number of papers related to interpretable or interactive ML divided by the total amount of ML papers for the given year.

Two queries were used in step (2), one for interpretable ML and another for interactive ML. The query for interpretable ML was as follows:

```
TAK=(``machine learning'' AND
(interpretable OR understandable OR
comprehensible OR explainable OR
intelligible OR transparent))
```

Where TAK stands for paper Title, Abstract or Keywords. The query for interactive ML was the following:

```
TAK=(``machine learning'' AND
(interactive OR ``human-in-the-loop''))
```

The query keywords were agreed upon by all authors and

were then used in the following databases: the ACM Digital Library, IEEE Xplore, Scopus, Science Direct and Web of Science. Figure 1 shows the results to these queries. Journals and conference papers from each database were sorted by relevance (as given by each database) and only the first 200 papers of each database were taken. Additionally, we crawled two web pages, NIPS[1] and PMLR[2], and ran the queries on the retrieved titles and abstracts – these web pages do not provide paper keywords. The resulting papers were ranked by their average number of citations per year and then reduced to only those with two or more average citations. Figure 2 shows a distribution of papers following this restriction: a total of 357 papers with 208 matching interpretable ML and 149 interactive.

From the total set of ranked papers, twenty six papers were surveyed by all authors in an individual manner, as an exercise for the development of the taxonomy. These papers were picked subjectively within the ranked list.

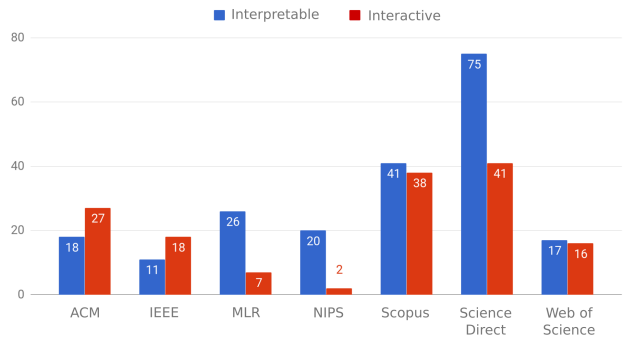


Figure 2: Number of papers per database, matching the queries for interpretable and interactive ML, with average citations per year equal or higher than two.

4 Taxonomy

The proposed taxonomy (Figure 3) is composed of six elements: Dataset, Optimizer, Model, Predictions, Evaluator and Goodness. Three of the elements – Optimizer, Model, and Evaluator – are based on Domingos's [2012] components of ML, whereas the other three – Dataset, Predictions and Goodness – have been added based on our observations from the surveyed papers. These observations are in line with the results reported in [Glauner *et al.*, 2017], where it is noted that a broader view on machine learning is needed, which includes not only the model but also the data, optimization techniques and evaluation metrics, a view that has, so far, been largely ignored in the literature. In Figure 3, white arrows represent inputs and black arrows outputs. Black boxes represent elements which produce an output given one or more inputs. The optimizer component (O), for example, has two inputs, a training dataset (X) and a model (M), and one output which is a new optimized model (M).

Each of the components in the taxonomy can potentially be enabled for user interpretation and/or interaction, as shown in the following subsections. The papers selected are not exclusive in the sense that they can only be used as illustrations for one of all the components shown in Figure 3, but can indeed showcase interpretability and interactivity within several of the components. Thus, the purpose of the examples is to describe in more details our view upon the taxonomy and its classifications.

4.1 Dataset

Dataset (X) regards training data, test data, validation data and/or input data for prediction or classification. It is expected to interpret and interact with the selection of parameters/features, its predictive probabilities, and the quality of the classified/predicted outcomes. It ultimately needs to support the users to make the next decision.

A solution which contributes to user **interpretation** of a dataset is performed by step by step user changes to the original feature values and allowing the users to compare/rank the models or classification/prediction results. The user needs to have the overview of the parameters selected as well as the corresponding results since following up the incremental effects is crucial. In fact, interpretation of a dataset is very closely connected to user interaction. A solution which contributes to user **interaction** with a dataset is one that lets the user have control of the penalizing/rewarding activities or which enables the user to simply change a value (e.g., feature weight) to see the corresponding classification results or predictions, mostly in real time. It is very related to user interpretation which guides the user in the model creation/update by interacting with the input data.

Two examples of research papers that contribute to the interaction with interpretation of predictions or classifications are [Krause *et al.*, 2016] and [Krause *et al.*, 2014].

In [Krause *et al.*, 2016], a tool is presented which detects diabetes and makes risk predictions by testing the patients' glucose measures. By tweaking features, they find the most impacting feature that brings a high-risk level of diabetes. The tool allows users to interpret how features affect the prediction with the help of visual representations to interact and compare. The interaction with the tool helps the user to diagnose the dependencies of features and to find the feature with the most impact.

[Krause *et al.*, 2014] bring insights to clinical researchers predicting patient outcomes by manipulating factors that affect a disease e.g., diabetes. They developed a visual analytics tool that enables interactive feature selection on high dimensional data with the ranking results of multiple feature selections, cross-validation folds, and classifiers. The predictive power is evaluated by ranking features (feature selection algorithms: information gain, Fisher score, odds ratio, relative risk) across multiple classification algorithms (tree, logistic regression, naive Bayesian, k-nearest neighbors) for the users to see the relevant features and compare the results.

4.2 Optimizer

The optimizer (O) – or optimization algorithm – is in charge of improving a model depending on the ML problem (e.g., classification, regression, clustering). To do so, it takes two inputs, a training data set X and a model M, and constructs an

[1]https://papers.nips.cc/ [Accessed 2018-05-15]
[2]http://proceedings.mlr.press/ [Accessed 2018-05-15]

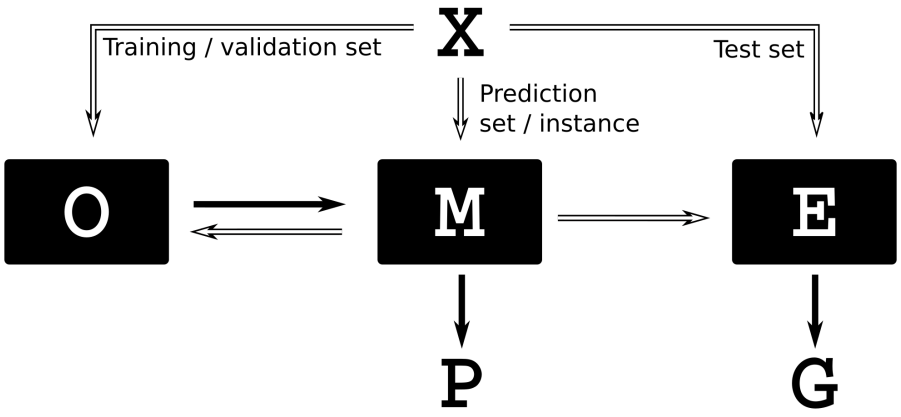


Figure 3: A taxonomy for interpretable and interactive Machine Learning. 'O' stands for optimizer, 'M' for model, 'E' for evaluator, 'P' for prediction, 'X' represents a dataset and 'G' the goodness of the model. Arrows depict inputs (white arrows) and outputs (black arrows). Each component can be potentially "opened" for user interpretation and interaction.

improved instance of M that better describes or generalizes X. Examples of optimizers are Linear Regression and Quasi-Newton methods such as DFP and BFGS.

Note that our definition is that of a component and not of a process. In that sense, an optimizer is not an optimization or a learning process. An optimizer is, however, an element of such a process. The distinction is important because elements of a process, and the interactions they can provide, are different. These processes are relevant to iiML but are outside of the scope of this paper. We dedicate, nevertheless, some space to the topic in Section 5.

A solution which contributes to user **interpretation** of an optimizer is one which discloses, in a *simple* human-readable manner, optimizer parameters (e.g., the learning rate or a distance function) and their impact to the output (i.e., the optimized model). A solution which contributes to optimizer **interaction** is one which allows users to change views of the optimizer, zoom in into its details, *tune* its parameters (e.g., change the learning rate or the distance function), or even change the optimizer itself.

Two examples of research papers that contribute to the interaction with, and interpretation of, an optimizer are [Holzinger *et al.*, 2016] and [Schreck *et al.*, 2009]. The former describes how a user can see and influence the behavior of the Ant Colony Optimization algorithm in the context of the Traveling Salesman Problem. Interpretation is facilitated through visual cues in a graph, with edges representing paths and their width the level of pheromones; interaction, on the other hand, is provided by allowing the user to manipulate pheromone levels. Schreck *et al.* [2009] describe a system which allows users to change the learning rate, as well as the neighborhood kernel, of a Self-Organizing Map for clustering trajectories. Such updates are then reflected on the new optimized versions of the model in the forthcoming iterations of the learning process.

4.3 Model

Learning algorithms are used to create a mathematical abstraction or generalization of data. This abstraction is called a model – or representation [Domingos, 2012] – and is represented by the 'M' box in Figure 3. Models are often implemented in a way that, given a new observation X, produces a classification or prediction P, e.g., given the profile of a client X, compute the risk of, e.g., giving him/her a bank credit (a classification P in the form of *low risk - high risk*).

A solution which contributes to user **interpretation** of a model is one which produces or wraps the mathematical abstraction – i.e., parameters, expressions, structure – in a format which facilitates human inspection and a human-readable explanation of its logic. Moreover, a solution which contributes to model **interaction** is one which allows users to change views of the model, get details on demand, and/or manipulate its inner elements (e.g., change parameters or its structure). Such solutions should support transparent interaction with humans without requiring that a user has expert knowledge of the ML techniques used. For example, a solution which computes and communicates ML results in ways that are compatible with the human decision-making process, and that can readily incorporate human experts' domain knowledge can be said to be interpretable and interactive.

Two examples of research papers that contribute to the interaction with, and interpretation of, a model are [Letham *et al.*, 2015] and [Hu *et al.*, 2014]. Letham *et al.* [2015] present a generative model called Bayesian Rule List which produces models in the form of small sets of *if... then* rules. Their contribution claims a balance between interpretability, accuracy and computational demand. Hu *et al.* [2014] developed a framework and a system for integrating user feedback into topic models in an interactive manner. After modeling a given number of topics 'M', their system allows users to modify them by adding, removing or increasing the relevance of words.

4.4 Prediction

The prediction component (P) of the taxonomy proposed regards the explanation of a prediction or classification produced by a model (M) to the human user as well as the possibility for the user refine it.

A solution which contributes to user **interpretation** of predictions or classifications is one which *explains* these result to the human user, i.e., why has X been classified as Y (and not Z)? As the actual features of the model can be difficult for a non-ML expert to interpret, the explanations generated need sometimes include other features than those used by the model for ensuring efficient human interpretation. A solution which contributes to user **interaction** with predictions or classifications is one which enables the user to investigate and tune how different parameter settings affect the resulting prediction/classification output, thus setting a foundation for increased knowledge of the workings of the model.

Two examples of research papers that contribute to the interaction with, and interpretation of, predictions or classifications are [Ribeiro *et al.*, 2016] and [Kulesza *et al.*, 2015]. In [Ribeiro *et al.*, 2016] an explanation technique is presented which explains the prediction of a classifier in an interpretable manner. For example, if applied in a medical scenario, the user is presented with a classification of the patient's illness/status together with the evidence for/against this classification. By inspecting these evidence, the users can use their expert knowledge of the domain to determine whether to trust the classification or not. [Ribeiro *et al.*, 2016] further argue that the explanations also can be used to select the most appropriate model for the problem at hand, by comparing the predictions of several models with the user's expert knowledge.

In [Kulesza *et al.*, 2015], an approach towards explainable ML is suggested, where the users are presented with explanations for the system's predictions to enable them to build mental models of the learning system as well as to interactively personalize it. The features of the classifier that are used to make the prediction are presented to the user, together with how each feature contributed to the prediction. Font size and color is used to convey this information, together with percentages of the likelihood of the prediction being correct. To correct the predictions made, the user is allowed to input or remove features from the explanation, which in turn will add or remove those features from the ML model's feature set. The users are also enabled to adjust the importance of the features in the explanation by increasing/decreasing the size of the feature in the interface, resulting in a higher/lower weight of the feature in the learning model.

4.5 Evaluator and Goodness

The evaluator (E) carries out the assessment of the performance of a model (M). Typically, traditional objective metrics from ML and Data Mining are used for this purpose, for instance, accuracy, precision, recall, squared error, f-score, information gain, etc. Such metrics are typically specialized to the type of machine learning problem and method used, i.e., clustering, classification, regression, etc. The input of the evaluator is usually the model itself and a test set, in order to assess the performance or "goodness" (G, output) of the model.

The evaluation of the model might be different from the overall performance of the ML-based system. Since we are considering interpretable and interactive ML solutions, the overall evaluation, and also the internal one, might include subjective metrics as well, commonly used in HCI, for example, usability evaluations (how long did users take to carry out certain tasks, were they successful, how many errors did they make, how many commands/features did they use, etc.). An example of model evaluation that includes subjective assessments is, for instance, presented in Amershi *et al.* [2010]. General challenges related to model evaluation are discussed in Fiebrink *et al.* [2011], where the authors conclude, among other issues, that exploratory evaluations of models can complement objective metrics in allowing users to evaluate models against a wide range of criteria.

A solution which contributes to user **interpretation** of an evaluator is one which allows the user to understand the evaluation, for example, showing the results of the performance of the predictions through visualizing the accuracy, error rate, etc. A solution which contributes to user **interaction** with an evaluator is one which supports user understanding and tuning of the evaluation process.

Two examples of research papers that contribute to the interaction with, and interpretation of an evaluator are [Bohanec *et al.*, 2017] and [Kapoor *et al.*, 2010], respectively. Bohanec *et al.* [2017] present a framework for explaining the results of classification models. According to our taxonomy, the evaluation carried out by Bohanec *et al.* is interpretable, since the explanations provided are claimed to support a better understanding of the classification accuracy of the models. Kapoor *et al.* [2010] present *ManiMatrix*, a system that support users in classification tasks using ML. The visualization of intermediate steps of the process supports and enhance the classification process, in some cases outperforming the highest automatic accuracy ever published for the problems in question. The evaluation presented in [Kapoor *et al.*, 2010] uses an interactive confusion matrix, which represents classification results by aggregating instances within a grid; each row in the matrix represents an instance's true class and each column an instance's predicted class. The users can specify interactively an increase or decrease in the tolerance for numbers of cases classified into each cell.

5 Discussion

The proposed taxonomy gives a detailed view of ML components and works as a reference for structuring how users can interpret and interact with each of them. The examples given illustrate some of the different ways of how the research community has contributed to iiML.

The taxonomy represents low-level components of ML. It is low-level for it does not explicitly depict higher processes such as, e.g., the machine learning process or – at an even higher level – the decision-making process. Such a fine-grained taxonomy can be challenging to use, for it requires a deeper understanding of how ML systems are implemented. Some system implementations have layers in between the user and the ML components, as means to map and transform input from one end to the other. Additionally, a user interaction can, in cases, trigger chains of transformations across all ML components. Telling where the impact of the user feedback will take effect is not always straightforward.

The research community has contributed to user involve-

ment in higher level processes such as the algorithmic learning process. Such a process can be found implicitly in the *Optimizer-Model* loop. Human interpretation at this level can involve process-wise elements such as [Mühlbacher *et al.*, 2014]: aliveness, i.e., status of the learning process (e.g., "learning in progress" or "has failed"); and progress, e.g., estimated remaining time. Human interaction, on the other hand, may translate to user control over the process, e.g., cancel execution, prioritize work [Mühlbacher *et al.*, 2014]. Contributions on this level, and others, are relevant to iiML but are not in the scope of this paper.

We envision research challenges at the current stage of our work. Challenges with the dataset component (X) may be given by the complexity of the data itself. Even relatively small datasets can be very complex to understand and handle. Interacting and interpreting graph or image data will probably prove more difficult than tabular numeric data. A challenge with optimizers (O) is their disassociation to the knowledge domain of the task, that is, they reside in a mathematical realm to which users might not relate to. The same might be said about models (M), although in their case certain formats have proved to be more interpretable than others (e.g. decision trees in contrast to neural networks). A recurrent challenge with models is the trade-off between interpretability and accuracy.

Papers reviewed in this domain that present evaluations (E) use either traditional objective performance methods and metrics from ML/DM or subjective assessments from HCI. Few present a combination of both strategies. We think that there is a lack of methods and metrics that can assess the overall performance of human-machine collaboration, which go beyond the evaluation of specific components of the whole system. Therefore, evaluation and metrics that combine both strategies are needed in the future.

6 Conclusions and Future Work

This paper presented a taxonomy for classifying and describing papers in the area of iiML. The aim of such taxonomy is to structure the literature found in this interdisciplinary area up to now and investigate through examples how ML can become more interpretable and interactive. The proposed taxonomy has six components, Dataset, Optimizer, Model, Prediction, Evaluator, and Goodness. We provided a description of each along with relevant papers to illustrate their role in iiML.

We provided brief descriptions of the different terminologies used under the scope of iiML, but believe there is a need for an agreed upon terminology to be used by HCI and ML practitioners, in order to better structure future work within the area. By exemplifying how the human user can be incorporated into the various ML components, we hope that our work can inspire practitioners within the field to develop highly functioning iiML systems where the strengths of the humans and the machines are efficiently exploited.

Through our work, we have identified an increasing interest of iiML as a research field, yet also the lack of examples of ML systems where the user is incorporated in all of the ML components outlined. With this paper, where we have identified low-level components on iiML, we hope to highlight this challenge and the need for addressing it in future work within the field.

As future work, we intend to investigate and describe higher-level processes of iiML, such as the learning process. A better understanding of how these processes are implemented, and how they are tailored for user interpretation and interaction, can prove useful for better structuring current research and for outlining feature research. We expect the findings to be helpful for building a complete taxonomy for iiML.

Acknowledgments

We would like to thank Nikolas Huhnstock and Niclas Ståhl at the Univeristy of Skövde for fruitful discussions regarding the proposed taxonomy.

References

[Abdul *et al.*, 2018] Ashraf Abdul, Jo Vermeulen, Danding Wang, Brian Y Lim, and Mohan Kankanhalli. Trends and trajectories for explainable, accountable and intelligible systems: An hci research agenda. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, page 582. ACM, 2018.

[Amershi *et al.*, 2010] Saleema Amershi, James Fogarty, Ashish Kapoor, and Desney Tan. Examining multiple potential models in end-user interactive concept learning. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1357–1360. ACM, 2010.

[Amershi *et al.*, 2013] Saleema Amershi, Maya Cakmak, W Bradley Knox, Todd Kulesza, and Tessa Lau. IUI workshop on interactive machine learning. In *Proceedings of the Companion Publication of the International Conference on Intelligent User Interfaces*, pages 121–124. ACM, 2013.

[Bibal and Frénay, 2016] Adrien Bibal and Benoît Frénay. Interpretability of machine learning models and representations: An introduction. In *Proceedings on ESANN*, pages 77–82, 2016.

[Bohanec *et al.*, 2017] Marko Bohanec, Marko Robnik-Šikonja, and Mirjana Kljajić Borštnar. Decision-making framework with double-loop learning through interpretable black-box machine learning models. *Industrial Management & Data Systems*, 117(7):1389–1406, 2017.

[Boukhelifa *et al.*, 2018] Nadia Boukhelifa, Anastasia Bezerianos, and Evelyne Lutton. Evaluation of interactive machine learning systems. *arXiv preprint arXiv:1801.07964*, 2018.

[Chiticariu *et al.*, 2015] Laura Chiticariu, Yunyao Li, and Fred Reiss. Transparent machine learning for information extraction. *EMNLP (tutorial)*, 2015.

[Chuang *et al.*, 2012] Jason Chuang, Daniel Ramage, Christopher Manning, and Jeffrey Heer. Interpretation and trust: Designing model-driven visualizations for text analysis. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 443–452. ACM, 2012.

[Domingos, 2012] Pedro Domingos. A few useful things to know about machine learning. *Commun. ACM*, 55(10):78–87, October 2012.

[Fails and Olsen, 2003] Jerry Alan Fails and Dan R. Olsen, Jr. Interactive machine learning. In *Proceedings of the 8th International Conference on Intelligent User Interfaces*, IUI '03, pages 39–45, New York, NY, USA, 2003. ACM.

[Fiebrink *et al.*, 2011] Rebecca Fiebrink, Perry R. Cook, and Dan Trueman. Human model evaluation in interactive supervised learning. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pages 147–156, New York, NY, USA, 2011. ACM.

[Glauner *et al.*, 2017] P. Glauner, M. Du, V. Paraschiv, A. Boytsov, I. Lopez Andrade, J. Meira, P. Valtchev, and R. State. The top 10 topics in machine learning revisited: A quantitative meta-study. In *Proceedings of the 25th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN 2017)*, pages 299–304, 2017.

[Holzinger *et al.*, 2016] Andreas Holzinger, Markus Plass, Katharina Holzinger, Gloria Cerasela Crişan, Camelia-M. Pintea, and Vasile Palade. *Towards interactive Machine Learning (iML): Applying Ant Colony Algorithms to Solve the Traveling Salesman Problem with the Human-in-the-Loop Approach*, pages 81–95. Springer International Publishing, Cham, 2016.

[Hu *et al.*, 2014] Yuening Hu, Jordan Boyd-Graber, Brianna Satinoff, and Alison Smith. Interactive topic modeling. *Machine Learning*, 95(3):423–469, Jun 2014.

[Kapoor *et al.*, 2010] Ashish Kapoor, Bongshin Lee, Desney Tan, and Eric Horvitz. Interactive optimization for steering machine classification. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1343–1352. ACM, 2010.

[Krause *et al.*, 2014] J. Krause, A. Perer, and E. Bertini. Infuse: Interactive feature selection for predictive modeling of high dimensional data. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):1614–1623, Dec 2014.

[Krause *et al.*, 2016] Josua Krause, Adam Perer, and Kenney Ng. Interacting with predictions: Visual inspection of black-box machine learning models. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16, pages 5686–5697, New York, NY, USA, 2016. ACM.

[Kulesza *et al.*, 2015] Todd Kulesza, Margaret Burnett, Weng-Keen Wong, and Simone Stumpf. Principles of explanatory debugging to personalize interactive machine learning. In *Proceedings of the 20th International Conference on Intelligent User Interfaces*, pages 126–137. ACM, 2015.

[Lee *et al.*, 2017] Tak Yeon Lee, Alison Smith, Kevin Seppi, Niklas Elmqvist, Jordan Boyd-Graber, and Leah Findlater. The human touch: How non-expert users perceive, interpret, and fix topic models. *International Journal of Human-Computer Studies*, 105(Supplement C):28 – 42, 2017.

[Letham *et al.*, 2015] Benjamin Letham, Cynthia Rudin, Tyler H McCormick, David Madigan, et al. Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model. *The Annals of Applied Statistics*, 9(3):1350–1371, 2015.

[Lim *et al.*, 2009] Brian Y. Lim, Anind K. Dey, and Daniel Avrahami. Why and why not explanations improve the intelligibility of context-aware intelligent systems. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, pages 2119–2128, New York, NY, USA, 2009. ACM.

[Lipton, 2016] Zachary C. Lipton. The mythos of model interpretability. In *ICML Workshop on Human Interpretability of Machine Learning*, 2016.

[Michalski, 1983] Ryszard S. Michalski. A theory and methodology of inductive learning. *Artificial Intelligence*, 20(2):111 – 161, 1983.

[Mühlbacher *et al.*, 2014] T. Mühlbacher, H. Piringer, S. Gratzl, M. Sedlmair, and M. Streit. Opening the black box: Strategies for increased user involvement in existing algorithm implementations. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):1643–1652, Dec 2014.

[Piltaver *et al.*, 2016] Rok Piltaver, Mitja Luštrek, Matjaž Gams, and Sanda Martinčić-Ipšić. What makes classification trees comprehensible? *Expert Systems with Applications*, 62:333 – 346, 2016.

[Ribeiro *et al.*, 2016] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1135–1144. ACM, 2016.

[Schreck *et al.*, 2009] Tobias Schreck, Jürgen Bernard, Tatiana von Landesberger, and Jörn Kohlhammer. Visual cluster analysis of trajectory data with interactive kohonen maps. *Information Visualization*, 8(1):14–29, 2009.

[Stumpf *et al.*, 2009] Simone Stumpf, Vidya Rajaram, Lida Li, Weng-Keen Wong, Margaret Burnett, Thomas Dietterich, Erin Sullivan, and Jonathan Herlocker. Interacting meaningfully with machine learning systems: Three experiments. *International Journal of Human-Computer Studies*, 67(8):639 – 662, 2009.

[Zhou, 2005] Zhi-Hua Zhou. Comprehensibility of data mining algorithms. *Encyclopedia of Data Warehousing and Mining*, pages 190–195, 2005.

Explainable Security

Luca Viganò, Daniele Magazzeni
Department of Informatics
King's College London, UK
luca.vigano@kcl.ac.uk, daniele.magazzeni@kcl.ac.uk

Abstract

The Defense Advanced Research Projects Agency (DARPA) recently launched the Explainable Artificial Intelligence (XAI) program that aims to create a suite of new AI techniques that enable end users to understand, appropriately trust, and effectively manage the emerging generation of AI systems.

In this paper, inspired by DARPA's XAI program, we propose a new paradigm in security research: Explainable Security (XSec). We discuss the "Six Ws" of XSec (Who? What? Where? When? Why? and How?) and argue that XSec has unique and complex characteristics: XSec involves several different stakeholders (i.e., the system's developers, analysts, users and attackers) and is multi-faceted by nature (as it requires reasoning about system model, threat model and properties of security, privacy and trust as well as about concrete attacks, vulnerabilities and countermeasures). We define a roadmap for XSec that identifies several possible research directions.

1 Introduction

The security of information, data, processes, software, protocols, computers, networks and systems is notoriously a challenging problem (and very often an undecidable one). Security is difficult. It is difficult to achieve, to reason about, to apply, to understand, to teach. It is difficult to *explain*.

The Defense Advanced Research Projects Agency (DARPA) recently launched the

Explainable Artificial Intelligence (XAI)

program that aims to create a suite of new AI techniques that enable end users to understand, appropriately trust, and effectively manage the emerging generation of AI systems. Some research on explainable AI had already been published before DARPA's program (e.g., [Swartout *et al.*, 1991; Ye and Johnson, 1995; Aldeco-Pérez and Moreau, 2008; Aldeco-Pérez and Moreau, 2010; Bidot *et al.*, 2010; Sohrabi *et al.*, 2011; Seegebarth *et al.*, 2012; Papadimitriou *et al.*, 2012; Gedikli *et al.*, 2014]), but XAI encouraged a large number of researchers to take up this challenge. In the last couple of years, several publications have appeared that investigate how to explain the different areas of AI, such as *machine learning* [Hendricks *et al.*, 2016], *recommender systems* [Muhammad *et al.*, 2016], *robotics* and *autonomous systems* [Rosenthal *et al.*, 2016; Sheh, 2017; Hayes and Shah, 2017], *constraint reasoning* [Freuder, 2017] and *planning* [Chakraborti *et al.*, 2017; Fox *et al.*, 2017; Langley *et al.*, 2017].[1]

In this paper, inspired by the XAI program, we propose a new paradigm in security research:

Explainable Security (XSec).

Some pioneering works on explaining security have focused on *security for relational databases* [Bender *et al.*, 2014] and on *explanation and trust* [Pieters, 2011]. In [Bender *et al.*, 2014], Bender, Kot and Gehrke propose a new model in which policy decisions are explainable. In this model, instead of simply rejecting an unauthorized query by a principal, the system provides the principal with a concise explanation of why the query was rejected and what additional permissions the principal would need to be granted for a successful execution. The principal can then refine the query or request additional permissions based on the explanation provided.

In [Pieters, 2011], Pieters investigates the relation between explanation and trust, focusing in particular on expert systems and e-voting systems. Pieters observes that

> *In artificial intelligence, explanations are usually provided by the system itself. In information security, explanations are provided by the designers. Nonetheless, in* both *artificial intelligence and information security, the role of explanations consists for a major part of acquiring and maintaining the trust of the user of the system.*

He discusses how explanations are required for trust:

> *Here, the question is how it is possible to communicate the analysis that experts have made of a security-sensitive system to the public. Why is it secure? Or, more appropriately: How is it secure?*

Explanations are thus

> *thought to bridge the gap between 'actual security' and 'perceived security'.*

[1] See also [Miller, 2017] and the other papers listed at `http://home.earthlink.net/~dwaha/research/meetings/faim18-xai/`.

Pieters also discusses two main goals that an explanation may have: *transparency* (e.g., to allow users to understand what the designers have done to protect them) and *justification* (e.g., offering reasons for an action). He contrasts *explanation-for-trust* (i.e., explanation of how a system works, by revealing details of its *internal* operations) with *explanation-for-confidence* (i.e., explanation to make the user feel comfortable in using the system, by providing information on its *external* communications).

We argue that XSec is a difficult problem and that it has unique and complex characteristics. In fact, XSec is more complex than what is discussed by Bender et al. and by Pieters; it is also more complex than *usable security* [Payne and Edwards, 2008; Wash and Zurko, 2017], *security awareness* [Banfield, 2016; Yildirim, 2016] and *security economics* [Anderson, 2018]. This is because XSec involves several different stakeholders (i.e., the system's developers, analysts, users and attackers) and is multi-faceted by nature (as it requires reasoning about system model, threat model and properties of security, privacy and trust as well as about concrete attacks, vulnerabilities and countermeasures). XSec is thus an exciting novel paradigm that requires a full-fledged and heterogeneous research program to be realized. In the following, we define a roadmap that identifies several possible research directions. To describe the challenges of XSec and how they could be tackled, we proceed by discussing the "Six Ws" of XSec summarized in Figure 1: Who? What? Where? When? Why? and How?

2 Who?

Consider a generic *system*, where we use here "system" to refer to a generic process, software, protocol, computer, network, cyber-physical system, critical infrastructure, etc. that processes information/data whose security must be protected, where "security" similarly generically refers to one or more of the security properties of interest, including confidentiality, integrity, availability, authentication, authorization, non-repudiation, accountability, unobservability, privacy, etc.

The *dramatis personae* of XSec are:

- the *designer* (and/or *developer*) of the system, who has designed (and/or developed) the system to guarantee a number of specified security properties;
- the *user* (and/or *client*) of the system, who can typically be assumed to be an honest non-expert who might commit mistakes that make the system vulnerable;
- the *attacker* (or more *attackers*) of the system, who searches for and exploits vulnerabilities of the system for reasons or profit, fame, reward, etc.;
- the *analyst* of the system, who carries out a semi-formal or formal analysis of the system at design time (or based on the specification of a deployed system) or tests the system at runtime (e.g., using penetration testing, vulnerability-based testing or model-based testing);
- the *defender* of the system, who attempts to protect the system, e.g., by monitoring the activities of the system and reacting to the attacker's actions.

For some situations, the recipient of the explanation will be an agent rather than a human, and we can then contrast *internal explanations* (designed for software agents) and *external explanations* (designed for humans, which is what XAI research typically focuses on).

Some of the above roles might actually be played by the same "principal" (agent or human), as the designer might for instance act also as analyst or defender, the analyst might also provide immediate defense and the attacker might be a user of the system. The literature is full of examples of vulnerabilities caused by mistakes by designers or users, along with details of the corresponding attacks. Some of these attacks could have been prevented by better explanations. In fact,

> all of these roles might require explanations or need to act as explainer,

For instance,

- a designer and/or an analyst might need to explain to the user how to interact with the system, why the system is secure and why it carries out a particular action (in line with what is discussed in [Bender *et al.*, 2014; Pieters, 2011]);
- a user or client might need to explain to the designer or the analyst how he expects the system to behave and how they typically interact with the system, to allow the designer to elicit the requirements for building the system in the first place and to allow the analyst to validate the security of user interactions;
- an attacker might need to explain the attack strategy to his accomplices so that they can attack in coordination, or he might have used a complex penetration testing tool to test the system for vulnerabilities and now needs the tool to explain to him the attack trace (or attack plan or strategy) that has been identified so that he can carry out the attack for real;
- an analyst might need to explain to the designer how to improve the system's security or to the defender how and what to defend;
- a defender might similarly need to understand possible attack traces in order to take action against them as well as explain to the users how they should behave to protect the system and themselves.

In addition to this, it is also necessary to tackle the research question of what actually constitutes a good (and secure) explanation, as we discuss in more detail in the following sections.

Before we do so, let us consider a concrete example that arises from the observation that when dealing with sensitive data, classical authentication solutions based on username-password pairs are not enough. The "General Data Protection Regulation" [European Commission, 2016] mandates that specific security measures must be implemented, including *multi-factor authentication (MFA)*, an authentication solution that aims to augment the security of the basic username-password authentication by exploiting two or more authentication factors (see, e.g., [Sciarretta *et al.*, 2018]). In [European Central Bank, 2014], MFA is for instance defined as:

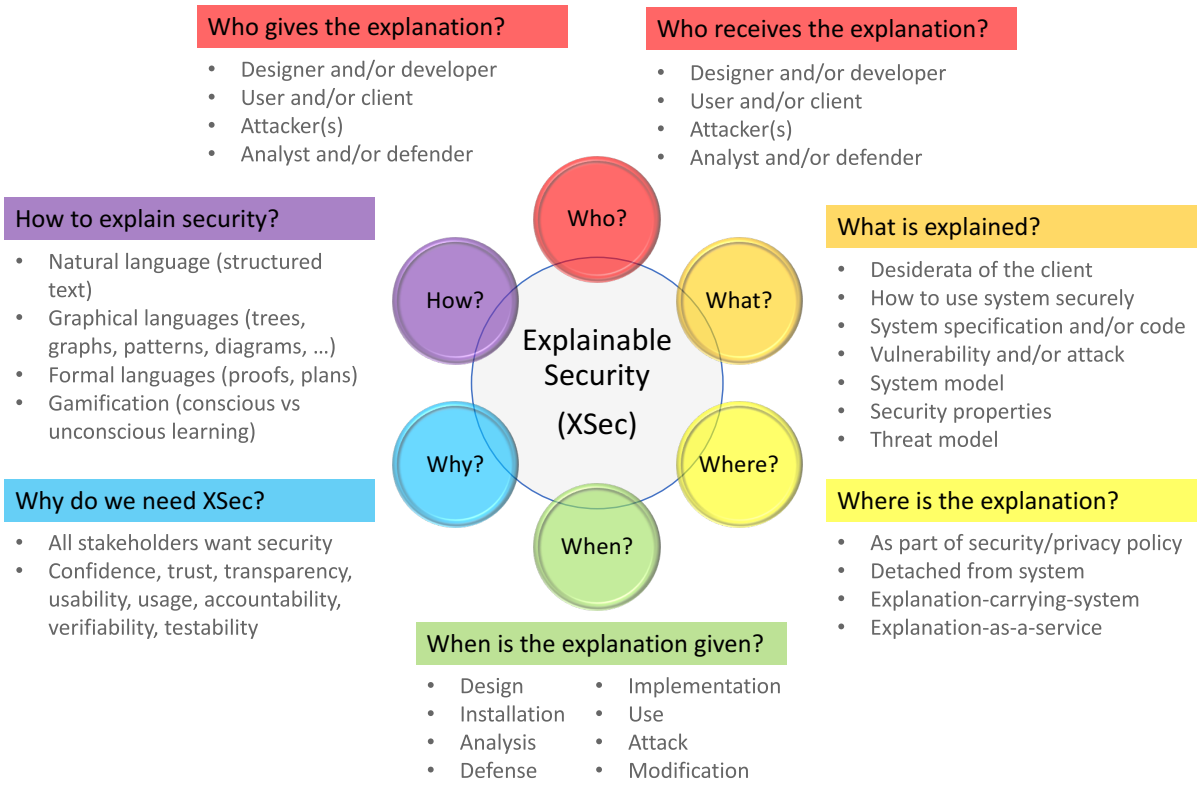


Figure 1: The Six Ws of Explainable Security

> *"a procedure based on the use of two or more of the following elements — categorised as knowledge, ownership and inherence: i) something only the user knows, e.g., static password, code, personal identification number; ii) something only the user possesses, e.g., token, smart card, mobile phone; iii) something the user is, e.g. a biometric characteristic, such as a fingerprint. In addition, the elements selected must be mutually independent [...] at least one of the elements should be non-reusable and non-replicable".*

The underlying idea is that the more factors are used during the authentication process, the more confidence a service has that the user is correctly identified. This is the basic explanation provided by the designer to the user to justify a more involved authentication that the user might perceive as cumbersome. However, the user might also need to be told that choosing a weak password is a bad idea even in the case of MFA. Two attacker accomplices who carry out a coordinated attack against the two components of MFA might need to explain their sub-attack to each other to ensure their ultimate success. The analyst who has discovered an attack to the MFA system might need to explain to the designer why the attack succeeeded and how to patch it. The analyst/defender might also need to explain to the users why they should, e.g., abandon the use of one of the elements they had been using so far and switch to using another pair of elements; for instance, because the new password that a user has chosen is too weak and thus easily guessable, or because the device on which the user is trying to authenticate does not include a biometric reader.

There are thus many things to explain by/to many different stakeholders, which is one of the main reasons why XSec, even in the case of a relatively simple example such as MFA, is a challenging endeavor. In the following, we discuss the five remaining Ws, although we have already anticipated much of the discussion here (which is somewhat unavoidable given that the Ws are not independent but quite deeply intertwined).

3 What?

It is not enough to explain the system in a generic way. First of all, the different stakeholders will need explanations at different levels of detail and with different aims:

- designers/developers will need an explanation of the desiderata of the client that is detailed enough for them to be able to realize the system in a satisfactory and secure way (e.g., if the client wants a system that replaces passwords with face recognition);
- non-expert users will need an explanation that increases their confidence and trust, and that also teaches them how to use the system correctly and securely (e.g., if passcodes are used as back-up in case face recognition fails as in the iPhone X, then the user should be made aware that the passcode ought to be strong enough and

not guessable such as a date of birth or a phone number, since otherwise an attacker who steals the iPhone X will obviously fail face recognition but the iPhone X will allow him to get access by guessing the passcode);

- analysts will need access to the system's specification or to the system's code in order to be able to create a model to analyze or to be able to generate and execute test cases;
- designers/developers/defenders will need an explanation of a vulnerability and related attacks in order to implement patches or defenses;
- attackers will need an explanation of how to exploit possible vulnerabilities, of why their attacks failed and of the implications of new security techniques on their attack strategies.

Second, several different "things" will need to be explained, including:

- the *system* and the *system model* used for design, implementation and analysis, e.g., the model of how MFA actually works;
- the *security properties* that the system should guarantee. e.g., the authentication provided by MFA can be used as a basis to provide authorization, integrity, confidentiality, non-repudiation and so on;
- the *threat model* that has been considered by designers, developers and analysts, highlighting, in particular, the fact that a system might be secure against one threat model but insecure against another (e.g., a system might be secure against an outside attacker but insecure against insider attacks) or the fact that the successful MFA of a user won't prevent the system from being attacked when that user turns out to be malignant and reveals, say, trade secrets of the company he works for;
- the actual *vulnerability* and related *attack* that has been discovered will need to be explained to the attacker (especially when the attacker used a tool to search for an attack and now needs to carry out it concretely), along with the costs and benefits of the attack;
- the possible *countermeasures* for the discovered *vulnerability* and related *attack* will need to be explained (along with the vulnerability and the attack) to the honest stakeholders who will need to understand the attack's impact, its risk and the mitigation strategies.

To that end, it will be helpful to answer a number of questions, including:

- What is actually secure? Which parts of the system? Which security properties are guaranteed and for how long? (For instance, authentication is typically granted for a session, which expires after some amount of time.) Which features are insecure and why and how can they be attacked? Are there different levels of security (e.g., for users with different rights)?
- What is the threat model considered? Does it include insiders and outsiders? Who are the potential attackers and what do they want? Why do they want it?
- How does the attack look like and how "difficult" is it? How expensive is it? (It does not make sense to use a one million dollar machine to mount an attack with a loot of a few thousand dollars.) How long will the attack take? (If students try to steal the questions of their next exam but their attack takes so long that they get hold of the questions only after the exam has been given by the professors, then there is actually no point.) This requires reasoning quantitatively about the economics of the attack (including costs, performance, time) but also about the trade-off between attacking and the risk of being discovered.
- Under which assumptions and conditions is the system assumed to be operating securely or has been proved to be secure? For instance, many security analyses (e.g., of protocols or web applications) typically assume a *Dolev-Yao-style attacker* [Dolev and Yao, 1983] who controls the network but cannot break cryptography, which is quite a strong assumption to make as cryptography might indeed be broken (by classical computers and even more so by quantum computers if and when they will be realized in their full capacity); on the other hand, relaxing this assumption and considering an attacker who might be able to break cryptography typically complicates the analysis (the problem is undecidable anyway) and the ability of an analyst to prove security guarantees.
- What are the *legal implications* of the explanation? Is the explanation "binding"? This would require, for instance, explaining how the system works and what is expected of the user, possibly including a digital signature to acknowledge the receipt and understanding of the explanation. In case of an attack, this would also require explaining what happened and why, and what countermeasures can be taken (and by whom).

4 Where?

We have already addressed the question of which "parts" of the system need to be explained in the "What?" section. Now we focus briefly on the question of where the explanations should be made available. A number of different options are available here, including the following four main ones:

- One could include the explanations to the users as part of the security/privacy policy, but it is well known that users typically ignore the policy and scroll down as quickly as possible so that they can get on with their interaction with the system.
- One could completely detach the explanation from the system, e.g., by making it available on a different webpage, but it is unclear to us if and how the relevant stakeholders will be made aware of where to find the explanation and whether they will decide to trust it.
- One could consider a sort of *explainable security as a service*, where stakeholders interact with an expert system to obtain and/or provide explanations.
- One could proceed in the style of *proof-carrying code* [Necula, 1997], "appending" a possibly digitally-

signed explanation to the system to achieve a *security-explanation-carrying-system*. We believe that this is the most promising direction, but it will of course require considerable work to protect the explanation from attacks and actually explain to the stakeholder how they can access it and make use of it.

5 When?

We want Explainable Security and we want it now! Jokes aside, the many vulnerabilities that are reported daily, including some of our most widespread and supposedly secure systems (consider, e.g., recent attacks against: TLS; PGP; processors; dropbox, one drive, iCloud and other cloud systems; biometric authentication systems; e-commerce and e-banking systems; e-voting systems, etc.), are witness to the fact that security is indeed difficult to achieve (which is why security has been and still is one of the hottest research topics) but also that in many cases security systems are difficult to explain to the different stakeholders.

We need to explain security when the system is

- designed,
- implemented,
- deployed and installed,
- used,
- analyzed,
- attacked,
- defended,
- modified,

and possibly even when the system is decommissioned and replaced, so that the different stakeholders understand why this decision was taken and how the new system will improve over the old one.

In particular, explanations will need to be defined and provided at *design time* (when the system is developed) but also at *runtime* (when the system is running). For the runtime case, think, e.g., of a critical system, critical infrastructure or cyber-physical system such as a nuclear power plant in which a supervisor is in charge of setting high/low security levels and of intervening in the case of an ongoing attack to estimate the success chances of the attack, understand its impact on the system and adopt possible countermeasures (see, e.g., [Lanotte *et al.*, 2017]). The attack could have disastrous consequences (e.g., manipulating the SCADA and PLC systems of a power plant as the Stuxnet Worm [Falliere *et al.*, 2011]) or (appear to) be non-threatening as it manipulates sensors and actuators of the system but without bringing them outside of their tolerance zone so that the supervisor actually decides not to intervene.

6 Why?

This is easiest question to answer: because all the different stakeholders of a system want it to be secure (well, with the exception of the attacker, of course). Explanations will help increase confidence, trust, transparency, usability and concrete usage (in the sense that users will be more keen to adopt the system), accountability, verifiability and testability.

7 How?

As we already remarked above, the different stakeholders will need explanations at different levels of detail and with different aims, and these explanations will need to be comprehensible, timely and accurate (among other properties). The explanations will need to be written in a language (and with a description strategy) suitable for the intended audience, including

- *natural language* (used to produce informal but possibly structured text written in English or any other language understandable by the audience);
- *graphical languages* such as explanation trees, attack trees, attack-defense trees, attack graphs, attack patterns, message-sequence charts, use case diagrams ...;
- *formal languages* including proofs and plans;
- games that have been produced as the result of a *gamification* process to teach users how to interact with a system (although one could actually object that such games often provide for some "unconscious learning" in which the user learns how to interact but without really understanding why).

It should also be investigated whether one learns more by seeing a proof of the security of the system or by being shown an attack against an insecure system. Both are of course useful, but they explain different things in a different way, and they can both be traced back to the question asked by Pieters [2011] of "how it is possible to communicate the analysis that experts have made of a security-sensitive system to the public".

It will also be necessary to evaluate security explanations originating from applications of XAI and other areas of computer science, and possibly even social sciences, psychology and other disciplines. Careful subject studies will need to be designed to assess measurement categories such as: a priori measures of explanation quality, user satisfaction, mental model understanding, and user-machine task performance.

Moreover, the explanation processes will themselves have to be designed properly, tested thoroughly and deployed correctly, and it will be useful to investigate the trade-off between such explanation processes and the security threats.

We expect that many of the How? questions posed in the specific case of security will actually be answerable by suitably adapting and extending the techniques and tools that have been and are being developed for XAI. Still, we conclude the discussion of the Ws by considering again the claim that we made above that XSec has unique and complex characteristics, and is more challenging than the pioneering research on explanations in security [Bender *et al.*, 2014; Pieters, 2011]. Let us illustrate this by an example that shows that XSec calls for proper extensions of the research on XAI and for novel investigations.

In Explainable Planning [Fox *et al.*, 2017], one of the questions the planner should answer is why things cannot be done and why and how one needs to replan. Similarly, in the model of Bender et al., the relational database system provides the principal with a concise explanation of why the query was rejected and what additional permissions the principal would

need to be granted for a successful execution. If one considers a more general security system, however, such an explanation might make the system less secure! This is because the explanation itself might reveal security-sensitive information.

For instance, the attacker might not know whether a certain person is indeed a user of the system: trying to login pretending to be that user and being told that the user does not exist, or that the password is wrong, or that the user needs more privileges to be able to carry out some specific action already constitutes a leak of information.[2] Hence, explanations need to be "relativized" and in some cases made less "powerful" by withholding certain details. But a less powerful explanation is essentially an incomplete explanation, which will be ignored or not fully achieve its purpose. The quest for a reasonable trade-off thus makes XSec particularly challenging.

8 Conclusions

We see XSec as a new paradigm that brings together many hot topics and current trends but also indicates the need of exploring unchartered territory. This paper has just skimmed the surface of XSec by pointing out some of the main objectives and challenges, and defining a roadmap that identifies several possible research directions.

We have already begun a more detailed investigation of the different techniques and tools that can be exploited or need to be developed to answer the questions posed by the different Ws, as well as formalizing the relationships and interdependences between the Ws. More specifically, we expect that, using the growing amount of results on XAI together with the pioneering research on explanations in security and trust as a stepping stone, we will soon be able to provide some concrete answers. To that end, we will also take inspiration from research that have attempted to "bridge the gap" between different research questions and different communities, such as the work of Hoffman on planning for penetration testing [2015].

This is the beginning of a beautiful friendship between explanations and security, and we plan to be part of the fellowship that nurtures this friendship.

Acknowledgments

We thank David W. Aha and the anonymous reviewers of the "IJCAI/ECAI 2018 Workshop on Explainable Artificial Intelligence (XAI)" for their very useful comments and suggestions.

References

[Aldeco-Pérez and Moreau, 2008] Rocío Aldeco-Pérez and Luc Moreau. Provenance-Based Auditing of Private Data Use. In *Visions of Computer Science*, pages 141–152. British Computer Society, 2008.

[Aldeco-Pérez and Moreau, 2010] Rocío Aldeco-Pérez and Luc Moreau. A Provenance-Based Compliance Framework. In *Proceedings of FIS*, 2010.

[Anderson, 2018] Ross Anderson. Economics and Security Resource Page, 2018. `http://www.cl.cam.ac.uk/~rja14/econsec.html`.

[Banfield, 2016] James Michael Banfield. *A Study of Information Security Awareness Program Effectiveness in Predicting End-User Security Behavior*. PhD thesis, Eastern Michigan University, 2016.

[Bender *et al.*, 2014] Gabriel Bender, Łucia Kot, and Johannes Gehrke. Explainable Security for Relational Databases. In *Proceedings of SIGMOD*. 2014.

[Bidot *et al.*, 2010] Julien Bidot, Susanne Biundo, Tobias Heinroth, Wolfgang Minker, Florian Nothdurft, and Bernd Schattenberg. Verbal Plan Explanations for Hybrid Planning. In *Proceedings of MKWI*, 2010.

[Chakraborti *et al.*, 2017] T. Chakraborti, S. Sreedharan, Y. Zhang, and S. Kambhampati. Plan explanations as model reconciliation: Moving beyond explanation as soliloquy. In *Proceedings of IJCAI*, 2017.

[Dolev and Yao, 1983] Danny Dolev and Andrew Yao. On the Security of Public-Key Protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983.

[European Central Bank, 2014] European Central Bank. Final guidelines on the security of Internet payments. `https://www.eba.europa.eu/documents/10180/934179/EBA-GL-2014-12+%28Guidelines+on+the+security+of+internet+payments%29.pdf/f27bf266-580a-4ad0-aaec-59ce52286af0`, 2014.

[European Commission, 2016] European Commission. Regulation EU 2016/679 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation). `http://www.eugdpr.org`, 2016.

[Falliere *et al.*, 2011] N. Falliere, L. Murchu, and E. Chien. W32.Stuxnet Dossier, 2011.

[Fox *et al.*, 2017] Maria Fox, Derek Long, and Daniele Magazzeni. Explainable Planning. In *Proceedings of IJCAI Workshop on Explainable Planning*, 2017.

[Freuder, 2017] E. Freuder. Explaining ourselves: Human-aware constraint reasoning. In *Proceedings of AAAI*, 2017.

[Gedikli *et al.*, 2014] F. Gedikli, D. Jannach, and M. Ge. How Should I Explain? A Comparison of Different Explanation Types for Recommender Systems. *International Journal of Human–Computer Studies*, 72(4):367–382, 2014.

[Hayes and Shah, 2017] Bradley Hayes and Julie A. Shah. Improving robot controller transparency through autonomous policy explanation. In *Proceedings of HRI*, pages 303–312, 2017.

[2] As another example, consider argument passing within GET protocols in HTTP requests, which basically allow the attacker to understand which is the name of the server's variable that handles the request, thus facilitating the attack. POST protocols are to be preferred, but they can't be used always.

[Hendricks *et al.*, 2016] Lisa Anne Hendricks, Zeynep Akata, Marcus Rohrbach, Jeff Donahue, Bernt Schiele, and Trevor Darrell. Generating visual explanations. In *Proceedings of ECCV*, pages 3–19, 2016.

[Hoffmann, 2015] Jörg Hoffmann. Simulated Penetration Testing: From "Dijkstra" to "Turing Test++". In *Proceedings of ICAPS*, 2015.

[Langley *et al.*, 2017] Pat Langley, Ben Meadows, Mohan Sridharan, and Dongkyu Choi. Explainable Agency for Intelligent Autonomous Systems. In *Proceedings of AAAI*, 2017.

[Lanotte *et al.*, 2017] Ruggero Lanotte, Massimo Merro, Riccardo Muradore, and Luca Viganò. A Formal Approach to Cyber-Physical Attacks. In *Proceedings of CSF*, 2017.

[Miller, 2017] Tim Miller. Explanation in Artificial Intelligence: Insights from the Social Sciences. *CoRR*, abs/1706.07269, 2017.

[Muhammad *et al.*, 2016] Khalil Muhammad, Aonghus Lawlor, and Barry Smyth. Explanation-based Ranking in Opinionated Recommender Systems. In *Proceedings of AICS*, 2016.

[Necula, 1997] George C. Necula. Proof-Carrying Code. In *Proceedings of POPL*, 1997.

[Papadimitriou *et al.*, 2012] A. Papadimitriou, P. Symeonidis, and Y. Manolopoulos. A generalized taxonomy of explanations styles for traditional and social recommender systems. *Data Min Knowl Disc*, 24(3):555–583, 2012.

[Payne and Edwards, 2008] Bryan D. Payne and W. Keith Edwards. A brief introduction to usable security. *IEEE Internet Computing*, 12(3):13–21, 2008.

[Pieters, 2011] Wolter Pieters. Explanation and trust: What to tell the user in security and AI? *Ethics Inf Technol*, 13:53–64, 2011.

[Rosenthal *et al.*, 2016] Stephanie Rosenthal, Sai P. Selvaraj, and Manuela M. Veloso. Verbalization: Narration of Autonomous Robot Experience. In *Proceedings of IJCAI*, 2016.

[Sciarretta *et al.*, 2018] Giada Sciarretta, Roberto Carbone, Silvio Ranise, and Luca Viganò. Design, Formal Specification and Analysis of Multi-Factor Authentication Solutions with a Single Sign-On Experience. In *Proceedings of POST*, 2018.

[Seegebarth *et al.*, 2012] Bastian Seegebarth, Felix Müller, Bernd Schattenberg, and Susanne Biundo. Making Hybrid Plans More Clear to Human Users - A Formal Approach for Generating Sound Explanations. In *Proceedings of ICAPS*, 2012.

[Sheh, 2017] R.K. Sheh. "Why did you do that?" Explainable intelligent robots. In *Proceedings of AAAI Workshop on Human-Aware Artificial Intelligence*, 2017.

[Sohrabi *et al.*, 2011] Shirin Sohrabi, Jorge A. Baier, and Sheila A. McIlraith. Preferred Explanations: Theory and Generation via Planning. In *Proceedings of AAAI*, 2011.

[Swartout *et al.*, 1991] W. Swartout, C. Paris, and J. Moore. Explanations in knowledge systems: Design for explainable expert systems. *IEEE Expert*, 6(3):58–64, 1991.

[Wash and Zurko, 2017] R. Wash and M.E. Zurko. Usable Security. *IEEE Internet Computing*, 21(3):19–21, 2017.

[Ye and Johnson, 1995] L. Ye and P. Johnson. The impact of explanation facilities on user acceptance of expert systems advice. *MIS Quarterly*, 19(2):157–172, 1995.

[Yildirim, 2016] Ebru Yildirim. The importance of information security awareness for the success of business enterprises. In *Proceedings of AHFE*, pages 211–222. Springer, 2016.

Contrastive explanations for reinforcement learning in terms of expected consequences

J. van der Waa [1], **J. van Diggelen**[1], **K. van den Bosch**[1], **M. Neerincx**[1]
[1] Netherlands Organisation for Applied Scientific Research
jasper.vanderwaa@tno.nl

Abstract

Machine Learning models become increasingly proficient in complex tasks. However, even for experts in the field, it can be difficult to understand what the model learned. This hampers trust and acceptance, and it obstructs the possibility to correct the model. There is therefore a need for transparency of machine learning models. The development of transparent classification models has received much attention, but there are few developments for achieving transparent Reinforcement Learning (RL) models. In this study we propose a method that enables a RL agent to explain its behavior in terms of the expected consequences of state transitions and outcomes. First, we define a translation of states and actions to a description that is easier to understand for human users. Second, we developed a procedure that enables the agent to obtain the consequences of a single action, as well as its entire policy. The method calculates contrasts between the consequences of the user's query-derived policy, and of the learned policy of the agent. Third, a format for generating explanations was constructed. A pilot survey study was conducted to explore preferences of users for different explanation properties. Results indicate that human users tend to favor explanations about policy rather than about single actions.

1 Introduction

Complex machine learning (ML) models such as Deep Neural Networks (DNNs) and Support Vector Machines (SVMs) perform very well on a wide range of tasks [Lundberg and Lee, 2016], but their outcomes are often are often difficult to understand by humans [Weller, 2017]. Moreover, machine learning models cannot explain how they achieved their results. Even for experts in the field, it can be very difficult to understand what the model actually learned [Samek *et al.*, 2016]. To remedy this issue, the field of eXplainable Artificial Intelligence (XAI) studies how such complex but useful models can be made more understandable [Gunning, 2017].

Achieving transparency of ML models has multiple advantages [Weller, 2017]. For example, if a model designer knows why a model performs badly on some data, he or she can start a more informed process of resolving the performance issues [Kulesza *et al.*, 2015; Papernot and McDaniel, 2018]. However, even if a model has high performance, the users (typically non-experts in ML) would still like to know why it came to a certain output [Miller, 2017]. Especially in high-risk domains such as defense and health care, inappropriate trust in the output may cause substantial risks and problems [Lipton, 2016; Ribeiro *et al.*, 2016]. If a ML model fails to provide transparency, the user cannot safely rely on its outcomes, which hampers the model's applicability [Lipton, 2016]. If, however, a ML-model is able to explain its workings and outcomes satisfactorily to the user, then this would not only improve the user's trust; it would also be able to provide new insights to the user.

For the problem of classification, recent research has developed a number of promising methods that enable classification models to explain their output [Guidotti *et al.*, 2018]. Several of these methods prove to be model-independent in some way, allowing them to be applied on any existing ML classification model. However, for Reinforcement Learning (RL) models, there are relatively few methods available [Verma *et al.*, 2018; Shu *et al.*, 2017; Hein *et al.*, 2017]. The scarcity of methods that enable RL agents to explain their actions and policies towards humans severely hampers the practical applications of RL-models in this field. It also diminishes the, often highly rated, value of RL to Artificial Intelligence [Hein *et al.*, 2017; Gosavi, 2009]. Take for example a simple agent within a grid world that needs to reach a goal position while evading another actor who could cause termination as well as evading other static terminal states. The RL agent cannot easily explain why it takes the route it has learned as it only knows numerical rewards and its coordinates in the grid. The agent has no grounded knowledge about the 'evil actor' that tries to prevent it from reaching its goal nor has it knowledge of how certain actions will effect such grounded concepts. These state features and rewards are what drives the agent but do not lend themselves well for an explanations as they may not be grounded concepts nor do they offer a reason why the agent behaves a certain way.

Important pioneering work has been done by Hayes and Shah [Hayes and Shah, 2017]. They developed a method for eXplainable Reinforcement Learning (XRL) that can generate explanations about a learned policy in a way that is understandable to humans. Their method converts feature vectors to a list of predicates by using a set of binary classification models. This list of predicates is searched to find sub-sets that tend to co-occur with specific actions. The method provides information about which actions are performed when which state predicates are true. A method that uses the co-occurrence to generate explanations may be useful for small problems, but becomes less comprehensible in larger planning and control problems, because the overview of predicate and action combinations becomes too large. Also, the method addresses only *what* the agent does, and not *why* it acts as it does. In other words, the method presents the user with the correlations between states and the policy but it does not provide a motivation why that policy is used in terms of rewards, or state transitions.

This study proposes an approach to XRL that allows an agent to answer questions about its actions and policy in terms of their consequences. Other questions unique to RL are also possible, for example those that ask about the time it takes to obtain some goal or those about RL specific problems (loop behavior, lack of exploration or exploitation, etc.). However we believe that a non-expert in RL is mostly interested in the expected consequences of the agent's learned behavior and whether the agent finds these consequences good or bad. This information can be used as an argument why the agent behaves in some way. This would allow human users to gain insight in what information the agent can perceive from a state and which outcomes it expects from an action or state visit. Furthermore, to limit the amount of information of all consequences, our proposed method aims to support contrastive explanations [Miller, 2017]. Contrastive explanations are a way of answering causal 'why'-questions. In such questions, two potential items, the fact and foil, are compared to each other in terms of their causal effects on the world. Contrastive questions come natural between humans and offer an intuitive way of gathering motivations about why one performs a certain action instead of another [Miller, 2017]. In our case we allow the user to formulate a question of why the learned policy π_t (the 'fact') is used instead of some other policy π_f (the 'foil) that is of interest to the user. Furthermore, our proposed method translates the set of states and actions in a set of more descriptive state classes $\mathbf{C}$ and action outcomes $\mathbf{O}$ similar to that of [Hayes and Shah, 2017]. This allows the user to query the agent in a more natural way as well as receive more informative explanations as both refer to the same concepts instead of plain features. The translation of state features to more high-level concepts and actions in specific states to outcomes, is also done in the proposed algorithm of [Sherstov and Stone, 2005]. The translation in this algorithm was used to facilitate transfer learning within a single action over multiple tasks and domains. In our method we used it to create a user-interpretable variant of the underlying Markov Decision Problem (MDP).

For the purpose of implementation and evaluation of our proposed method, we performed a pilot study. In this study, a number of explanation examples were presented to participants to see which of their varying properties are preferred the most. One of the properties was to see whether the participants prefer explanations about the expected consequences of a single-action or the entire policy.

2 Approach for consequence-based explanations

The underlying Markov Decision Problem (MDP) of a RL agent consists of the tuple $\langle S, A, R, T, \lambda \rangle$. Here, S and A are the set of states (described by a feature vector) and actions respectively, $R : S \times A \rightarrow \mathbb{R}$ is the reward function and $T : S \times A \rightarrow Pr(S)$ the transition function that provides a probability distribution over states. Also, λ is the discount factor that governs how much of future rewards are taken into account by the agent. This tuple provides the required information to derive the consequences of the learned policy π_t or the foil policy π_f from the user's question. As one can use the transition function T to sample the effects of both π_t and π_f. In the case T is not explicit, one may use a separate ML model to learn it in addition to the actual agent. Through this simulation, one constructs a Markov Chain of state visits under each policy π_t and π_f and can present the difference to the user.

Through the simulation of future states with T, information can be gathered about state consequences. In turn, from the agent itself the state or state-action values for simulated state visits can be obtained to develop an explanation in terms of rewards. However, the issue with this approach is that the state features and rewards may not be easy to understand for a user as it would consist of possibly low-level concepts and numerical reward values or expected returns. To mitigate this issue we can apply a translation of the states and actions to a set of predefined state concepts and outcomes. These concepts can be designed to be more descriptive and informative for the potential user. A way to do this translation is by training a set of binary classifiers to recognize each outcome or state concept from the state features and taken action, a similar approach to the one from [Hayes and Shah, 2017]. Their training can occur during the exploratory learning process of the agent. This translation allows us to use the above described method of simulating consequences and transform the state features and results of actions to more user-interpretable concepts.

2.1 A user-interpretable MDP

The original set of states can be transformed to a more descriptive set $\mathbf{C}$ according to the function $\mathbf{k} : S \rightarrow \mathbf{C}$. This is similar to the approach of [Hayes and Shah, 2017] where $\mathbf{k}$ consists of a number of classifiers. Also, rewards can be explained in terms of a set of action outcomes $\mathbf{O}$ according to $\mathbf{t} : \mathbf{C} \times A \rightarrow Pr(\mathbf{O})$. This provides the results of an action in some state in terms of the concepts $\mathbf{O}$. For example, the outcomes that the developer had in mind when designing the reward function R. The transformation of states and actions in state classes and outcomes is adopted from the work of [Sherstov and Stone, 2005] where the transformations are used to allow for transfer learning in RL. Here however, we

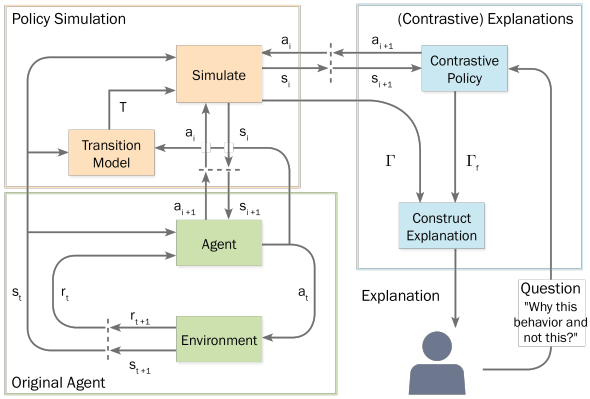


Figure 1: An overview of the proposed method, a dotted line represents a feedback loop. We assume a general reinforcement learning agent that acts upon a state s through some action a and receives a reward r. We train a transition model T that can be used to simulate the effect of actions on states. By repeatedly simulating a state s_i we can obtain the expected consequences γ of an entire policy. Also, the consequences of a contrastive policy consisting of an alternative courses of action a_f can be simulated with the same transition model T. Finally, in constructing the explanation we transform states and actions into user-interpretable concepts and construct an explanation that is contrastive.

use them as a translation towards a more user-interpretable representation of the actual MDP.

The result is the new MDP tuple $\langle S, A, R, T, \lambda, \mathbf{C}, \mathbf{O}, \mathbf{t}, \mathbf{k} \rangle$. An RL agent is still trained on S, A, R and T with λ independent of the descriptive sets $\mathbf{C}$ and $\mathbf{O}$ and functions $\mathbf{k}$ and $\mathbf{t}$. This makes the transformation independent of the RL algorithm used to train the agent. See Figure 1 for an overview of this approach.

As an example take the grid world illustrated in Figure 2 that shows an agent in a simple myopic navigation task. The states S are the (x, y) coordinates and the presence of a forest, monster or trap in adjacent tiles with $A = Up, Down, Left, Right$. R consists of a small transient penalty, a slightly larger penalty for tiles with a forest, a large penalty shared over all terminal states (traps or adjacent tiles to a monster) and a large positive reward for the finishing state. T is skewed towards the intended result with small probabilities for the other results if possible.

The state transformation $\mathbf{k}$ can consist out of a set of classifiers for the predicates whether the agent is next to a forest, a wall, a trap or monster, or in the forest. Applying $\mathbf{k}$ to some state $s \in S$ results in a Boolean vector $c \in C$ whose information can be used to construct an explanation in terms of the stated predicates. The similar outcome transformation $\mathbf{t}$ may predict the probability of the outcomes $\mathbf{O}$ given a state and action. In our example, $\mathbf{O}$ consists of whether the agent will be at the goal, in a trap, next to the monster or in the forest. Each outcome $\mathbf{o}$ can be flagged as being positive $\mathbf{o}^+$ or negative $\mathbf{o}^-$ purely such that they can be presented differently in the eventual explanation.

Given the above transformations we can simulate the next state of a single action a with T or even the entire chain of

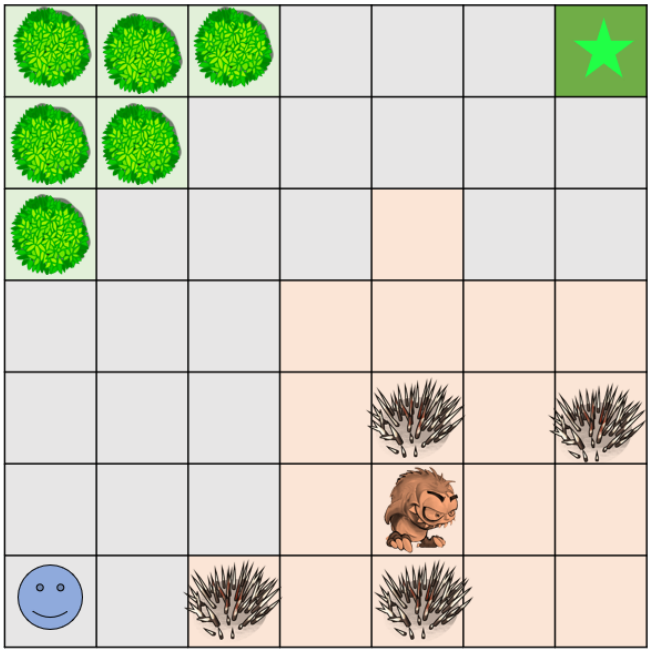


Figure 2: A simple RL problem where the agent has to navigate from the bottom left to the top right (goal) while evading traps, a monster and a forest. The agent terminates when in a tile with a trap or adjacent to the monster. The traps and the monster only occur in the red-shaded area and as soon as the agents enter this area the monster moves towards the agent.

actions and visited states given some policy π. These can then be transformed into state descriptions $\mathbf{C}$ and action outcomes $\mathbf{O}$ to form the basis of an explanation. As mentioned, humans usually ask for contrastive questions especially regarding their actions [Miller, 2017]. In the next section we propose a method of translating the foil in a contrastive question into a new policy.

2.2 Contrastive questions translated into value functions

A contrastive question consists of a fact and a foil, and its answer describes the contrast between the two from the fact's perspective [Miller, 2017]. In our case, the fact consists of the entire learned policy π_t, a single action from it $a_t = \pi_t(s_t)$ or any number of consecutive actions from π_t. We propose a method of how one can obtain a foil policy π_f based on the foil in the user's question. An example of such a question could be (framed within the case of Figure 2);

> "Why do you move up and then right *(fact)* instead of moving to the right until you hit a wall and then move up *(foil)*?"

The foil policy π_f is ultimately obtained by combining a state-action value function Q_I – that represents the user's preference for some actions according to his/her question – with the learned Q_t to obtain Q_f;

$$Q_f(s, a) = Q_t(s, a) + Q_I(s, a), \forall s, a \in S, A \qquad (1)$$

Each state-action value is of the form $Q : S \times A \rightarrow \mathbb{R}$.

Q_I only values the state-action pairs queried by the user. For instance, the Q_I of the above given user question can be based on the following reward scheme for all potentially simulated $s \in S$;

- The action a_f^1 = 'Right' receives a reward such that $Q_f(s, \text{Right}) > Q_t(s, \pi_t(s))$
- If 'RightWall' $\in \mathbf{k}(s)$
- Then the action a_f^2 = 'Up' receives a reward such that $Q_f(\cdot, \text{Up}) > Q_t(\cdot, \pi_t(s))$.

Given this reward scheme we can train Q_I and obtain Q_f according to equation 1. The state-action values Q_f can then be used to obtain the policy π_f using the original action selection mechanism of the agent. This results in a policy that tries to follow the queried policy as best as it can. The advantage of having π_f constructed from Q_f is that the agent is allowed to learn a different action then those in the user's question as long as the reward is higher in the long run (more user defined actions can be performed). Also, it allows for the simulation of the actual expected behavior of the agent as it is still based on the agent's action selection mechanism. This would both not be the case if we simply forced the agent to do exactly what the user stated.

The construction of Q_I is done through simulation with the help of the transition model T. The rewards that are given during the simulation are selected with Equation 1 in mind, as they need to eventually compensate for the originally learned action based on Q_t. Hence, the reward for each state and queried action is as follows;

$$R_I(s_i, a_f) = \frac{\lambda_f}{\lambda} w(s_i, s_t) \left[R(s_i, a_f) - R(s_i, a_t\right] (1 + \epsilon) \tag{2}$$

With $a_t = \pi_t(s_t)$ the originally learned action and w being a distance based weight;

$$w(s_i, s_t) = e^{-\left(\frac{d(s_i, s_t)}{\sigma}\right)^2} \tag{3}$$

First, s_i with $i \in \{t, t+1, ..., t+n\}$ is the i'th state in the simulation starting with s_t. a_f is the current foil action governed by the conveyed policy by the user. The fact that a_f is taken as the only rewarding action each time, greatly reduces the time needed to construct Q_I. Next, $w(s_i, s_t)$ is obtained from a Radial Basis Function (RBF) with a Gaussian kernel and distance function d. This RBF represents the exponential distance between our actual state s_t and the simulated state s_i. The Gaussian kernel is governed by the standard deviation σ and allows us to reduce the effects of Q_I as we get further from our actual state s_t. The ratio of discount factors $\frac{\lambda_f}{\lambda}$ allows for the compensation between the discount factor λ of the original agent and the potentially different factor λ_f for Q_I if we wish it to be more shortsighted. Finally, $[R(s_i, a_f) - R(s_i, a_t)] (1 + \epsilon)$ is the amount of reward that a_f needs such that $Q_I(s_i, a_f)\epsilon > Q(s_i, a_t)$. With $\epsilon > 0$ that determines how much more Q_I will prefer a_f over a_t.

The parameter n defines how many future state transitions we simulate and are used to retrieve Q_I. As a general rule $n \geq 3\sigma$ as at this point the Gaussian kernel will reduce the contribution of Q_I to near zero such that Q_f will resemble Q_t. Hence, by setting σ one can vary the number of states the foil policy should starting from s_t. Also, by setting ϵ the strength of how much each a_f should be preferred over a_t can be regulated. Finally, λ_f defines how shortsighted Q_I should be. If set to $\lambda_f = 0$, π_f will force the agent to perform a_f as long as s_i is not to distant from s_t. If set to values near one, π_f is allowed to take different actions as long as it results into more possibilities of performing a_f.

2.3 Generating explanations

At this point we have the user-interpretable MDP consisting of state concepts $\mathbf{C}$ and action outcomes $\mathbf{O}$ provided by their respective transformation function $\mathbf{k}$ and $\mathbf{t}$. Also, we have a definition of R_I that values the actions and/or states that are of interest by the user which can be used to train Q_I through simulation and obtain Q_f according to Equation 1. This provides us with the basis of obtaining the information needed to construct an explanation.

As mentioned before, the explanations are based on simulating the effects with T of π_t and that of π_f (if defined by the user). We can call T on the previous state s_{i-1} for some action $\pi(s_{i-1}$ to obtain s_i and repeat this until $i == n$. The result is a single sequence or trajectory of visited states and performed actions for any policy π starting from s_t;

$$\gamma(s_t, \pi) = \{(s_0, a_0), ..., (s_n, a_n) \mid T, \pi\} \tag{4}$$

If T is probabilistic, multiple simulations with the same policy and starting state may result in different trajectories. To obtain the most probable trajectory $\gamma(s_t, \pi)^*$ we can take the transition from T with the highest probability. Otherwise a Markov chain could be constructed instead of a single trajectory.

The next step is to transform each state and action pair in $\gamma(s_t, \pi)^*$ to the user-interpretable description with the functions $\mathbf{k}$ and $\mathbf{t}$;

$$Path(s_t, \pi) = \{(c_0, o_0), ..., (c_n, o_n)\}, \\ c_i = \mathbf{k}(s_i), o_i = \mathbf{t}(s_i, a_i), (s_i, a_i) \in \gamma(s_t, \pi)^* \tag{5}$$

From $Path(s_t, \pi_t)$ an explanation can be constructed about the state the agent will most likely visit and the action outcomes it will obtain. For example with the use of the following template;

> "For the next n actions I will mostly perform a. During these actions, I will come across situations with $\forall c \in Path(s_t, \pi_t)$. This will cause me $\forall o^+ \in Path(s_t, \pi_t)$ but also $\forall o^- \in Path(s_t, \pi_t)$".

Let a here be the action most common in $\gamma(s_t, \pi_t)$ and both o^+ and o^- the positive and negative action outcomes respectively. Since we have access to the entire simulation of π_f, a wide variety of explanations is possible. For instance we could also focus on the less common actions;

> "For the next n actions I will perform a_1 when in situations with $\forall c \in Path(s_t, \pi_t | \pi_t = a_1)$ and a_2 when in situations with $\forall c \in Path(s_t, \pi_t | \pi_t = a_2)$. These actions prevent me from $\forall o^+ \in Path(s_t, \pi_t)$ but also $\forall o^- \in Path(s_t, \pi_t)$".

A contrastive explanation given some question from the user that describes the foil policy π_f can be constructed in a similar manner but take the contrast. Given a foil we can focus on the differences between $Path(s_t, \pi_t)$ and $Path(s_t, \pi_f)$. This can be obtained by taking the relative complement $Path(s_t, \pi_t) \backslash Path(s_t, \pi_f)$; the set of expected unique consequences when behaving according to π_t and not π_f. A more extensive explanation can be given by taking the symmetric difference $Path(s_t, \pi_t) \triangle Path(s_t, \pi_f)$ to explain the unique differences between both policies.

3 User study

The above proposed method allows an RL agent to explain and motivate its behavior in terms of expected states and outcomes. It also enables the construction of contrastive explanations where any policy can be compared to the learned policy. This contrastive explanation is based on differences in expected outcomes between the compared policies.

We performed a small user study in which 82 participants were shown a number of exemplar explanations about the case shown in figure 2. These explanations addressed either the single next action or the policy. Both explanations can be generated by the above method by adjusting the Radial Basis Function weighting scheme and/or the foil's discount factor. Also, some example explanations were contrastive with only the second best action or policy, while others provided all consequences. Contrasts were determined using the relative complement between fact and foil. Whether the learned action or policy was treated as the fact or foil, was also systematically manipulated in this study.

We presented the developed exemplar explanations in pairs to the participants and asked them to select the explanation that helped them most to understand the agent's behavior. Afterwards we asked which of the following properties they used to assess their preference: long versus short explanations; explanations with ample information versus little information; explanations addressing actions versus those that address strategies (policies); and explanations addressing short-term consequences of actions versus explanations that address distant consequences of actions.

The results of the preferred factors are shown in Figure 3. This shows that the participants prefer explanations that address strategy and policy, and that provide ample information. We note here that, given the simple case from figure 2, participants may have considered an explanation addressing a single action only as trivial, because the optimal action was, in most cases, already evident to the user.

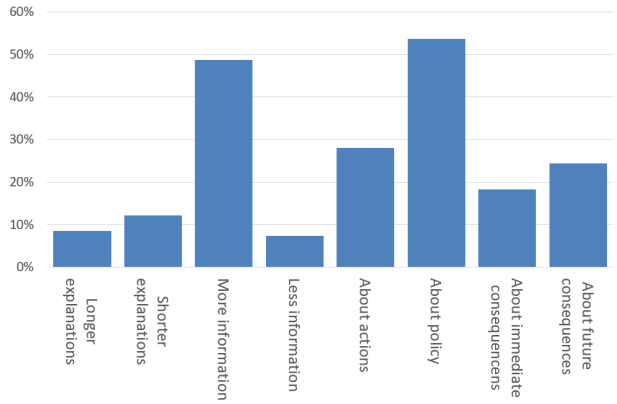


Figure 3: A plot depicting the percentage of participants (y-axis) for each explanation property (x-axis) that caused them to prefer some explanations over others. Answers of a total of 82 participants where gathered.

4 Conclusion

We proposed a method for a reinforcement learning (RL) agent to generate explanations for its actions and strategies. The explanations are based on the expected consequences of its policy. These consequences were obtained through simulation according to a (learned) state transition model. Since state features and numerical rewards do not lend themselves easily for an explanation that is informative to humans, we developed a framework that translates states and actions into user-interpretable concepts and outcomes.

We also proposed a method for converting the foil, –or policy of interest to the user–, of a contrastive 'why'-question about actions into a policy. This policy follows locally the user's query but gradually transgresses back towards the original learned policy. This policy favors the actions that are of interest to the user such that the agent tries to perform them as best as possible. How much these actions are favored compared to the originally learned action can be set with a single parameter.

Through running simulations for a given number steps of both the policy derived from the user's question and the actually learned policy, we were able to obtain expected consequences of each. From here, we were able to construct contrastive explanations: explanations addressing the consequences of the learned policy and what would be different if the derived policy would have been followed.

An online survey pilot study was conducted to explore which of several explanations are most preferred by human users. Results indicate that users prefer explanations about policies rather than about single actions.

Future work will focus on implementing the method on complex RL benchmarks to explore the scalability of this approach in realistic cases. This is important given the computational costs of simultaneously simulating the consequences of different policies in large state spaces. Also, we will explore more methods to construct our translation functions from states and actions to concepts and outcomes. A more extensive user study will be carried out to evaluate the instructional value of generated explanations in more detail, and to explore the relationship between explanations and users' trust in the agent's performance.

Acknowledgments

We would like to thank the reviewers for their time and effort in improving this paper. Also, we are grateful for the funding from the RVO Man Machine Teaming research project that made this research possible.

References

[Gosavi, 2009] Abhijit Gosavi. Reinforcement learning: A tutorial survey and recent advances. *INFORMS Journal on Computing*, 21(2):178–192, 2009.

[Guidotti *et al.*, 2018] Riccardo Guidotti, Anna Monreale, Franco Turini, Dino Pedreschi, and Fosca Giannotti. A survey of methods for explaining black box models. *arXiv preprint arXiv:1802.01933*, 2018.

[Gunning, 2017] David Gunning. Explainable artificial intelligence (xai). *Defense Advanced Research Projects Agency (DARPA), nd Web*, 2017.

[Hayes and Shah, 2017] Bradley Hayes and Julie A Shah. Improving robot controller transparency through autonomous policy explanation. In *Proceedings of the 2017 acm/ieee international conference on human-robot interaction*, pages 303–312. ACM, 2017.

[Hein *et al.*, 2017] Daniel Hein, Steffen Udluft, and Thomas A Runkler. Interpretable policies for reinforcement learning by genetic programming. *arXiv preprint arXiv:1712.04170*, 2017.

[Kulesza *et al.*, 2015] Todd Kulesza, Margaret Burnett, Weng-Keen Wong, and Simone Stumpf. Principles of explanatory debugging to personalize interactive machine learning. In *Proceedings of the 20th International Conference on Intelligent User Interfaces*, pages 126–137. ACM, 2015.

[Lipton, 2016] Zachary C Lipton. The mythos of model interpretability. *arXiv preprint arXiv:1606.03490*, 2016.

[Lundberg and Lee, 2016] Scott Lundberg and Su-In Lee. An unexpected unity among methods for interpreting model predictions. *arXiv preprint arXiv:1611.07478*, 2016.

[Miller, 2017] Tim Miller. Explanation in artificial intelligence: Insights from the social sciences. *arXiv preprint arXiv:1706.07269*, 2017.

[Papernot and McDaniel, 2018] Nicolas Papernot and Patrick McDaniel. Deep k-nearest neighbors: Towards confident, interpretable and robust deep learning. *arXiv preprint arXiv:1803.04765*, 2018.

[Ribeiro *et al.*, 2016] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1135–1144. ACM, 2016.

[Samek *et al.*, 2016] Wojciech Samek, Grégoire Montavon, Alexander Binder, Sebastian Lapuschkin, and Klaus-Robert Müller. Interpreting the predictions of complex ml models by layer-wise relevance propagation. *arXiv preprint arXiv:1611.08191*, 2016.

[Sherstov and Stone, 2005] Alexander A Sherstov and Peter Stone. Improving action selection in mdp's via knowledge transfer. In *AAAI*, volume 5, pages 1024–1029, 2005.

[Shu *et al.*, 2017] Tianmin Shu, Caiming Xiong, and Richard Socher. Hierarchical and interpretable skill acquisition in multi-task reinforcement learning. *arXiv preprint arXiv:1712.07294*, 2017.

[Verma *et al.*, 2018] Abhinav Verma, Vijayaraghavan Murali, Rishabh Singh, Pushmeet Kohli, and Swarat Chaudhuri. Programmatically interpretable reinforcement learning. *arXiv preprint arXiv:1804.02477*, 2018.

[Weller, 2017] Adrian Weller. Challenges for transparency. *arXiv preprint arXiv:1708.01870*, 2017.

Show Me What You've Learned: Applying Cooperative Machine Learning for the Semi-Automated Annotation of Social Signals

Johannes Wagner[1], **Tobias Baur**[1], **Dominik Schiller**[1], **Yue Zhang**[2], **Björn Schuller**[2], **Michel Valstar**[3], **Elisabeth André**[1]
[1] Augsburg University
[2] Imperial College London
[3] University of Nottingham
{wagner,baur,schiller,andre}@hcm-lab.de
{yue.zhang1,bjoern.schuller}@imperial.ac.uk
michel.valstar@nottingham.ac.uk

Abstract

In this paper we suggest the use of Cooperative Machine Learning (CML) to reduce manual labelling efforts while simultaneously generating an intuitive understanding of the learning process of a classification system. To this end, we introduce the open-source tool NOVA, which aims to combine human intelligence and machine learning to annotate social signals in large multi-modal corpora. NOVA features a semi-automated labelling process in which users are provided with immediate visual feedback on the predictions, which affords insights into the strengths and weaknesses of the underlying classification system. Following an interactive and exploratory workflow, the performance of the model can be improved by manual revision of the predictions, a process that uses confidence values to guide the inspection.

1 Introduction

In various research disciplines (Behavioural Psychology, Medicine, Anthropology, ...) the annotation of social behaviours is a common task. This process includes manually identifying relevant behaviour patterns in audio-visual material and assigning descriptive labels. Generally speaking, segments in the signals are labelled using sets of discrete classes or continuous scores, e. g., a certain type of gesture, a social situation (e. g., conflict), or the emotional state of a person. In Social Signal Processing (SSP), a subset of these events – the so called *social signals* – are used to augment the spoken part of a message with non-verbal information to enable a more natural human-computer interaction [Vinciarelli *et al.*, 2009][1]. To automatically detect social signals from raw sensory input (e. g., speech signals) it is common practice to apply machine learning (ML) techniques. That is, sensory input is transformed into a compact set of relevant features and a classifier is trained on manually labelled examples to optimise a learning function. Once trained, the classifier is used to automatically predict labels on unseen data.

However, since humans transmit non-verbal messages through a number of channels (voice, face, gestures, etc.) and due to the complex interplay between these channels, large amounts of annotated data are necessary to cover those phenomena. Therefore, the progress in the field of SSP is directly linked to the availability of large and well transcribed multi-modal databases rich of human behaviour under varying context and different environmental settings [Douglas-Cowie *et al.*, 2003]. Common challenges in creating such datasets lie in the high degree of naturalness required of the recording scenarios, how well one recording scenario generalises to other settings, the number of human raters needed to reach a consensus on labels, and of course the sheer amount of data. When one considers the many hours of labelled data that are required, gathering such large amounts of annotated training samples may seem like an infeasible task, with respect to time, cost and effort.

An obvious solution is to exploit computational power to accomplish some of the annotation work automatically. However, to ensure the quality of the predicted annotations this still requires human supervision to identify and correct errors. To keep the human effort as low as possible, it is useful to understand why a model makes wrong assumptions. Therefore, it is not only important to provide tools that ease the use of semi-automated labelling, but also to increase the transparency of the decision process (a non trivial task given that most modern classifiers come as black boxes). By visualising the predictions, for instance, even non ML experts get an idea about the strengths and weaknesses of the underlying classification model and can immediately decide which parts of a prediction are worth keeping. If a particular label is regularly missed, a user could actively provide more training examples for this phenomenon, or redesign the ML system to capture its relevant characteristics better. Ideally, the system even guides his or her attention towards parts where manual revision is necessary. Once an annotation has been revised, the model can be retrained to improve its performance for the next cycle. This procedure can be repeated until a desired

[1] To give an example of a social signal, think of a situation where we say something in a sarcastic voice to indicate that we actually mean the opposite.

performance is reached.

In this paper, we introduce an annotation tool called NOVA ((Non-)Verbal Annotator), which implements the described workflow that interactively incorporates the 'human in the loop'. In particular, NOVA offers an interface to acquire semi-automated annotations and provides visual feedback to inspect and correct machine-generated labels. In that sense, our work combines three hot topics of ML: *Explainable Artificial Intelligence*, as the transparency of the decision process is increased via visualisation of the predictions; *Active Learning*, since labels with low confidence are highlighted to guide the user towards relevant parts; and finally, *Interactive Machine Learning*, because human intelligence and machine power can cooperate and improve each other. We subsume our approach under the term *Cooperative Machine Learning* (see Section 3).

2 Related Work

Despite vast resources of raw data, nowadays pervasive in digital format and relatively easy and inexpensive to collect, e. g., from public resources such as social media, the problem of efficiently gathering relevant annotations still needs to be overcome. One approach is *Active Learning* (AL) [Zhu, 2005], a type of algorithm that interactively queries the user to manually label certain data points. The core idea of AL is to extract the most informative instances from a pool of unlabelled data based on a specific query strategy [Settles, 2010]. These selected instances are then passed to human annotators for labelling and a model is derived from this subset. This approach significantly reduces the labelling effort.

The work by Zhang *et al.* [2015c] takes the idea of AL a step forward and combines it with *Semi-Supervised Learning* (SSL) techniques to efficiently share the labelling work between a human annotator and a machine: a pre-existing classifier is used to predict labels for the unlabelled data. For each of those predictions a confidence level is calculated by the classifier. Only if this level falls below a certain threshold a human annotator is asked to revise the annotation. To further save labelling efforts, one can apply *Dynamic Active Learning* (DAL) by choosing the most reliable raters first [Zhang *et al.*, 2015b]. Zhang *et al.* [2015a] developed an agreement-based annotation technique that dynamically determines how many human annotators are required to label a selected instance. The technique considers individual rater reliability and inter-rater agreement to decide on a combination of raters to be allocated to an instance.

However, little emphasis is given to the question of how to assist users in the application of these techniques for the creation of their own corpora. While the benefits of integrating active learning with annotation tasks has been demonstrated in a variety of experiments, annotation tools that provide users with access to active learning techniques are rare. Recent developments for audio, image and video annotation that make use of active learning include CAMOMILE [Poignant *et al.*, 2016] and iHEARu-PLAY [Hantke *et al.*, 2015]. However, systematic studies focusing on the potential benefits of the active learning approach within the annotation environment from a user's point of view have been performed only rarely [Cheng and Bernstein, 2015; Kim and Pardo, 2017].

Interactive Machine Learning (IML) [Fails and Olsen, 2003; Amershi *et al.*, 2014] aims to involve users actively in the creation of models for recognition tasks. Most approaches integrate automated data analysis and interactive visualisation tools in order to enable users to inspect data, process features and tune models. An example includes ModelUI [Wagner *et al.*, 2010]. It presents users with a graphical user interface that allows them to test different ML algorithms on labelled data. Labels are acquired by stimuli which may include textual instructions, but also images or videos. Afterwards, users can review the recordings and correct the annotations.

Rosenthal and Dey [2010] investigated which kind of information should be provided to users in order to reduce annotation errors. They found out that contextual information and predictions of the learning algorithms were in particular useful for the annotation of activity data. In contrast, uncertainty information had no effect on the accuracy of the labels, but just indicated to the labellers that classification was difficult. Amershi *et al.* [2009] investigated how to empower users to select samples for training by appropriate visualisation techniques. They found that a representative overview of best and worst matching examples is of higher value than a set of high-certainty images and conjecture that high-certainty images do not provide much information to the learning processing due to their similarity to already labelled images. In another paper by Amershi *et al.* [2015], the authors suggest an interactive visualization technique in order to assess a models' performance. By sorting samples according to their prediction score, the user can directly retrieve additional information and annotate them for better performance tracking. This way, the tool allows users to monitor the performance of individual samples while the model is iteratively retrained.

In addition to presenting the outcome of a classification in a structured way, one may aim at opening up what is usually perceived as a 'black box': the classifier itself. *Explainable Artificial Intelligence* (XAI) deals with the problem of making AI decisions transparent and explainable [Samek *et al.*, 2017]. For example, displaying the closest match to an instance can be a simple, yet effective way to increase transparency of the classification process. However, in complex AI systems where reasoning is no longer based on instance matching such an approach is not sufficient. A detailed discussion of different theories of explanation is given in [Sørmo *et al.*, 2005].

Today, explainability becomes increasingly important as we rely more and more on AI in our everyday life. For instance, before trusting a 'black box' model in a mission-critical applications, e. g., the diagnosis of Pneumonia, we have to ensure that the prediction is not based on random factors such as overfitting and spurious correlation [Caruana *et al.*, 2015]. However, gaining insights into the inner workings of a classifier may not just prevent misapplication, but also bears potential for improving the system. Or as noted by Samek *et al.* [2017]: "the first step towards improving an AI system is to understand its weaknesses".

Summing up, it may be said that multiple studies empirically investigated the potential of novel techniques in order to minimise human labelling effort. In addition, some studies

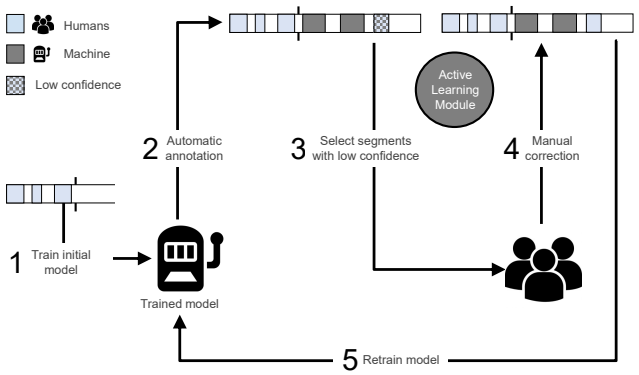


Figure 1: The scheme depicts the general idea behind Cooperative Machine Learning (CML): (1) An initial model is trained on partially labelled data. (2) The initial model is used to automatically predict unseen data. (3) Labels with a low confidence are selected and (4) manually revised. (5) The initial model is retrained with the revised data.

were conducted to actually label novel data, rather than test whether such method could save effort. Relatively little attention has been paid, however, to the question of how to make these techniques available to human labellers. There is a high demand for annotation tools that integrate ML techniques in order to reduce human effort – in particular in the area of social signal processing where human raters typically disagree on the labels [Lotfian and Busso, 2017].

3 Cooperative Machine Learning

In this paper, we subsume learning approaches that efficiently combine human intelligence with the machine's ability of rapid computation under the term *Cooperative Machine Learning* (CML) [Dong and Sun, 2003; Zhang *et al.*, 2015c]. In Figure 1, we illustrate our approach to CML, which creates a loop between a machine learned model and human annotators: an initial model is trained (1) and used to predict unseen data (2). An active learning module then decides which parts of the prediction are subject to manual revision by human annotators (3+4). Afterwards, the initial model is retrained using the revised data (5). Now the procedure is repeated until all data is annotated. By actively incorporating the user into the loop it becomes possible to interactively guide and improve the automatic predictions while simultaneously obtaining an intuition for the functionality of the classifier. In [Wagner *et al.*, 2018], we report an experiment that measures the increase in speed when the described CML strategy is applied within a realistic annotation task. Results showed that manual work was reduced by a factor of $\frac{5}{8}$.

However, the approach not only bears the potential to considerably cut down manual efforts, but also to come up with a better understanding of the capabilities of the classification system. For instance, the system may quickly learn to label some simple behaviours, which already facilitates the work load for human annotators at an early stage. Then, over time, it could learn to cope with more complex social signals as well, until at some point it is able to finish the task in a completely automatic manner. Such an iterative approach may even help bridging the gap between quantitative and qualitative coding, which still defines a great challenge in many fields in social science [Chen *et al.*, 2016].

To efficiently apply the described strategy, we would like to know the *sweet spot* for handing an annotation task over to the machine. On the one hand, if we do it too early, the model becomes unstable and predictions will be poor. On the other hand, if we annotate more data than necessary, we give away precious time. To avoid any of the described situations, we are interested in finding a good trade-off between machine performance and human effort. Unfortunately, we cannot easily guess what is the ideal moment to hand over the task to a machine. This is because the amount of training data that is required to build a robust model depends on a number of factors, such as the homogeneity of the data, the discrimination ability of the extracted features, the number of subjects and classes, and, not least, the complexity of the recognition problem. Alternatively, instead of trying to determine a sweet spot beforehand (and possibly miss it), we could iteratively test the applicability of the strategy and stop when the performance seems promising. Therefore, we opt to make the described strategy an integral part of a graphical interface (see Section 4). This allows annotators to visually examine the results at any time and to individually decide whether more labelling is required or not. This procedure can be further accelerated by providing visual feedback on the quality of the predictions. This way, annotators can concentrate on parts with *low confidence*, i. e., correcting only labels with a high uncertainty[2].

4 NOVA Tool

We will now introduce our novel annotation tool NOVA. The interface is inspired by existing software, such as EUDICO Linguistic Annotator (ELAN) [Wittenburg *et al.*, 2006] and Annotation of Video and Language (ANVIL) [Kipp, 2013], which offer layer-based tiers to insert time-anchored labelled segments – that is *discrete* annotations. In addition, NOVA also supports *continuous* annotations, which allow an observer to track the content of an audiovisual stimulus over time along one ore more dimensions – a feature inspired by software like GTRACE (General Trace) [Cowie *et al.*, 2012], CARMA (Continuous Affect Rating and Media Annotation) [Girard, 2014] and DARMA (Dual Axis Rating and Media Annotation) [Girard and Wright, 2016]. However, whereas the mentioned tools offer none or only little automation, NOVA has been advanced with features to create semi-automated annotations (see Section 3).

NOVA is open-source and available on Github: `https://github.com/hcmlab/nova`.

4.1 Main Interface

The NOVA user interface has been designed with a special focus on the annotation of long and continuous recordings involving multiple modalities and subjects. A screenshot of a loaded recording session is shown in Figure 2. On the top,

[2]Uncertainty can be derived from the distance a predicted sample has to the decision boundaries of the other classes. In regression problems, dropout can be used [Gal and Ghahramani, 2016]).

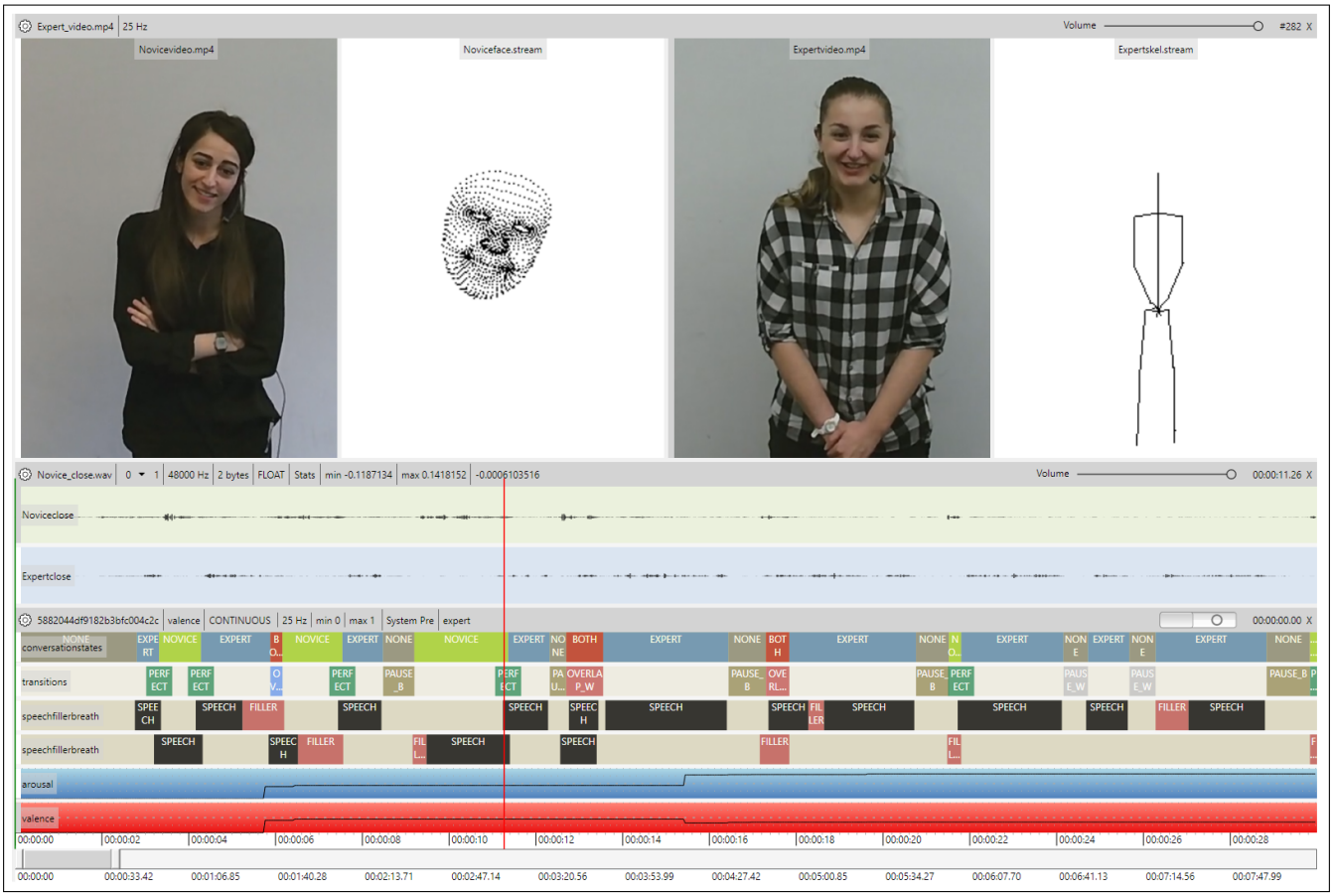


Figure 2: NOVA allows to visualise various media and signal types and supports different annotation schemes. From top downwards: full-body videos along with skeleton and face tracking, and audio streams of two persons during an interaction. In the lower part, several discrete and continuous annotation tiers are displayed. Annotations can be edited on a static fraction of the recording or interactively during playback.

several media tracks are visualised and ready for playback. Note that the number of tracks that can be displayed at the same time is not limited and various types of signals (video, audio, facial features, skeleton, depth images, etc.) are supported. In the lower part, we see multiple annotation tracks of different types (discrete, continuous and transcriptions) describing the visualised content.

To support a collaborative annotation process, NOVA maintains a database back-end, which allows users to load and save annotations from and to a MongoDB[3] running on a central server. This gives annotators the possibility to immediately commit changes and follow the annotation progress of others. Beside human annotators, a database may also be visited by one or more "machine users". Just like a human operator, they can create and access annotations. Hence, the database also functions as a mediator between human and machine. NOVA provides instruments to create and populate a database from scratch. At any time new annotators, schemes and additional sessions can be added.

NOVA provides several functions to process the annotations created by multiple human or machine annotators. For instance, statistical measures such as Cronbach's α or Cohen's κ can be applied to identify inter-rater agreement. Finally, multiple annotations can be merged to a Gold Standard. However, in the following we will concentrate on another feature of NOVA: the use of machine learning to support the user during the annotation process.

4.2 Machine Learning

For best possible performance, tasks related to machine learning (ML) are outsourced and executed in a background process. As backend we use our open-source Social Signal Interpretation (SSI) framework[4]. Since SSI is primarily designed to build online recognition systems, a trained model can be directly used to detect social cues in real-time [Wagner *et al.*, 2013].

A typical ML pipeline starts by preprocessing data to input data for the learning algorithm, a step known as *feature extraction*. An XML template structure is used to define extraction chains from individual SSI components. For instance, the following template extracts the commonly used Mel-frequency cepstral coefficients (MFCCs) from audio sig-

[3] `https://www.mongodb.com/`

[4] `http://openssi.net`

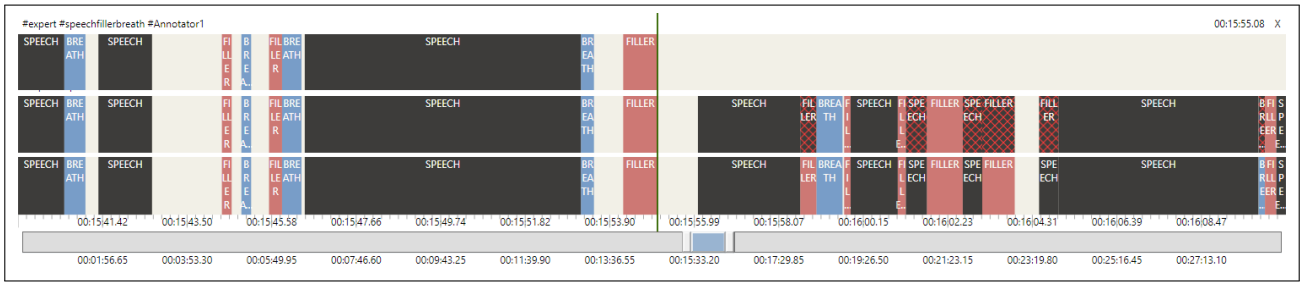


Figure 3: The upper tier shows a partly finished annotation. Machine learning is now used to predict the remaining part of the tier (middle), where segments with a low confidence are highlighted with a red pattern. The lower tier shows the final annotation after manual revision.

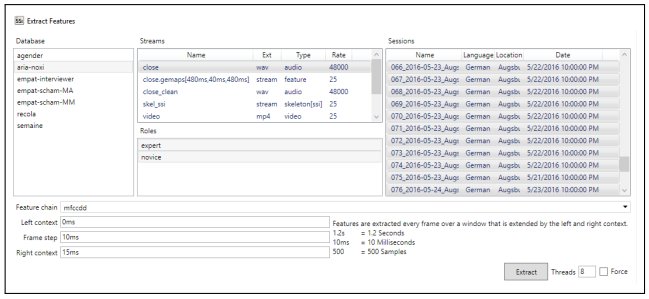


Figure 4: Feature extraction dialogue.

nals. It feeds the input signal through a pre-emphasis filter and afterwards extracts the features over a sliding window of 25 ms with a frame step of 10 ms.

```
<chain>
  <register name="audio"/> <!—load components—>
  <meta frameStep="10ms" rightContext="15ms"/>
  <filter> <!—apply filtering—>
    <item create="PreEmphasis"/>
  </filter>
  <feature> <!—extract features—>
    <item create="Mfcc" option="mfcc"/>
  </feature>
</chain>
```

A dialogue helps users to extract features by selecting an input stream and a number of sessions (see Figure 4). The result of the operation is stored as a new signal in the database. This way, feature streams can be reviewed in the GUI and accessed by all users. Based on the extracted features, a classifier can be trained. Again, templates are used to define classification schemes, e. g., :

```
<trainer>
  <register name="model"/>
  <meta balance="under"/> <!—apply under sampling—>
  <normalize>
    <item method="Scale"/> <!—scale the features—>
  </normalize> <!—Support Vector Machine—>
  <model create="SVM" option="svm"/>
</trainer>
```

To automatically finish an annotation, the user either selects a previously trained model or temporarily builds one using the labels on the current tier. An example before and after the completion is shown in Figure 3. Note that labels with a low confidence are highlighted with a pattern. This way, the annotator can immediately see how well the prediction worked. He or she can now either revert the operation or continue based on the automated generated annotation. At any time, usually after correcting a couple of false predictions or adding some missing labels, the procedure can be repeated. Over time, this should lead to increasingly stable predictions.

Summing up, the described methodology offers transparency from two directions. By observing the output of the classifier, the user can assess its performance and also trace how it changes with new input. In addition, visualising the input to the classifier (raw media or feature streams) can provide hints why a prediction was successful in one place but failed in another. For instance, the user may find out that predictions were wrong due to failure of the tracking algorithm. This way, users also learn in which situations they can trust the model.

Note that users can extend NOVA's ML tools by simply adding new templates. SSI supports a variety of features sets for different types of signals. For instance, it allows to extract a large number of audio parameters based on the widely used OPENSMILE toolkit [Eyben *et al.*, 2013]. For the computation of facial points and action units from video streams, the OPENFACE tool by Baltrušaitis *et al.* [2016] has been integrated. In terms of classification models, SSI supports (among others) Google's neural network framework TENSORFLOW[5] or the popular THEANO[6] library.

5 Conclusion

The goal of the presented work is to foster the application of *Cooperative Machine Learning* (CML) strategies to support the annotation of social signals in large multi-modal databases. Well described corpora that are rich of human behaviour are needed in a number of disciplines, such as Social Signal Processing and Behavioural Psychology. However, populating captured user data with adequate descriptions can be an extremely exhausting and time-consuming task. To this end, we have presented a novel annotation tool NOVA. It allows to distribute annotation tasks among multiple human raters and offers an interface to ML algorithms for semi-automated annotation.

The core idea of the presented work is to create a loop, in which humans start solving a task (here labelling social signals), and over time, a machine learns to automatically com-

[5]https://www.tensorflow.org/
[6]https://github.com/Theano/

plete the task. In conventional approaches, this involves at least two parties: an end-user, who has knowledge about the domain, and a machine learning practitioner, who can cope with the learning system. However, to make the process more rapid and focused, our tool combines a traditional annotation interface with techniques for automated labelling that can be applied out of the box requiring no knowledge on ML. For an optimised workflow, coders have the possibility to individually decide when and how to use them in the labelling process. Further, to assess the reliability of automatic predictions immediate visual feedback is provided, which gives annotators the chance to gain insights into the ML model and adapt their strategies at times. By interactively guiding and improving automatic predictions, an efficient integration of human expert knowledge and rapid mechanical computation is achieved.

Our experiences with NOVA show that CML strategies not only have the potential to speed up coding, but can also have a positive influence coding quality. Because of the preciseness machine-aided techniques introduce into the coding process, the level-of-detail is improved while at the same time human efforts are reduced. However, while a machine is able to annotate social signals much faster and more consistently than humans can do, human raters still bring a better understanding for the application in which the models to be trained will eventually be applied. Furthermore, human raters do not just look at the behaviours to be labelled, but also reason about the context in which they occur [Baur *et al.*, 2017]. Being presented with the results of an automated labelling process might influence human labellers in a positive manner. Nevertheless, one should be aware of the risk that a machine-like style of annotation might not always result in better systems. This is in particular true when social signals are analysed where raters usually disagree on the labels and no objective ground truth can be established. In order to benefit from the complementary skills of machines and human raters, annotation tools like NOVA are needed that aim for a smooth integration of human intelligence and resources.

In the future, we aim at further improving the explanation capabilities of the system by providing more information about the inner workings of the classifiers. This, for instance, could be achieved by adopting explanation approaches like the LIME-System of Ribeiro *et al.* [2016] or the Explicable-Boundary-Tree-Explainer by Wu *et al.* [2018]. The idea here is to not only visualize final predictions, but also disclose what has lead to a specific decision. We believe that this way, human resources could be applied even more effectively, which may further shorten the time it takes to achieve a stable classification performance.

Acknowledgements

This work is Funded by European Union Horizon 2020 research and innovation programme, grant agreement No. 645378 (ARIA-Valuspa).

References

Saleema Amershi, James Fogarty, Ashish Kapoor, and Desney S. Tan. Overview based example selection in end user interactive concept learning. In *Proceedings of the 22nd Annual ACM Symposium on User Interface Software and Technology, Victoria, BC, Canada, October 4-7, 2009*, pages 247–256, 2009.

Saleema Amershi, Maya Cakmak, W. Bradley Knox, and Todd Kulesza. Power to the people: The role of humans in interactive machine learning. *AI Magazine*, 35(4):105–120, 2014.

Saleema Amershi, Max Chickering, Steven M. Drucker, Bongshin Lee, Patrice Simard, and Jina Suh. Modeltracker: Redesigning performance analysis tools for machine learning. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pages 337–346. ACM, 2015.

Tadas Baltrušaitis, Peter Robinson, and Louis-Philippe Morency. Openface: an open source facial behavior analysis toolkit. In *Applications of Computer Vision (WACV), 2016 IEEE Winter Conference on*, pages 1–10. IEEE, 2016.

Tobias Baur, Dominik Schiller, and Elisabeth André. Modeling user's social attitude in a conversational system. In Marko Tkalcic, Berardina De Carolis, Marco de Gemmis, Ante Odic, and Andrej Kosir, editors, *Emotions and Personality in Personalized Services - Models, Evaluation and Applications*, Human-Computer Interaction Series, pages 181–199. Springer, 2017.

Rich Caruana, Yin Lou, Johannes Gehrke, Paul Koch, Marc Sturm, and Noemie Elhadad. Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, pages 1721–1730, New York, NY, USA, 2015. ACM.

Nan-Chen Chen, Rafal Kocielnik, Margaret Drouhard, Vanessa Peña-Araya, Jina Suh, Keting Cen, Xiangyi Zheng, and Cecilia R. Aragon. Challenges of applying machine learning to qualitative coding. In *CHI 2016 workshop on Human Centred Machine Learning*, 2016.

Justin Cheng and Michael S. Bernstein. Flock: Hybrid crowd-machine learning classifiers. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing*, pages 600–611. ACM, 2015.

Roddy Cowie, Gary McKeown, and Ellen Douglas-Cowie. Tracing emotion: An overview. *International Journal of Synthetic Emotions (IJSE)*, 3(1):1–17, January 2012.

Miaobo Dong and Zengqi Sun. On human machine cooperative learning control. In *Proceedings of the 2003 IEEE International Symposium on Intelligent Control*, pages 81–86, Oct 2003.

Ellen Douglas-Cowie, Nick Campbell, Roddy Cowie, and Peter Roach. Emotional speech: Towards a new generation of databases. *Speech Communication*, 40(c):33–60, 2003.

Florian Eyben, Felix Weninger, Florian Gross, and Björn Schuller. Recent developments in opensmile, the munich open-source multimedia feature extractor. In *Proceedings*

of the 21st ACM International Conference on Multimedia, MM '13, pages 835–838, New York, NY, USA, 2013. ACM.

Jerry Alan Fails and Dan R. Olsen, Jr. Interactive machine learning. In *Proceedings of the 8th International Conference on Intelligent User Interfaces*, IUI '03, pages 39–45, New York, NY, USA, 2003. ACM.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, pages 1050–1059, 2016.

Jeffrey M. Girard and Aidan G C Wright. DARMA: Dual Axis Rating and Media Annotation. *Manuscript submitted for publication*, 2016.

Jeffrey M. Girard. Carma: Software for continuous affect rating and media annotation. *Journal of Open Research Software*, 2(1):e5, 2014.

Simone Hantke, Florian Eyben, Tobias Appel, and Björn Schuller. ihearu-play: Introducing a game for crowdsourced data collection for affective computing. In *Affective Computing and Intelligent Interaction (ACII), 2015 International Conference on*, pages 891–897. IEEE, 2015.

Bongjun Kim and Bryan Pardo. I-sed: An interactive sound event detector. In *Proceedings of the 22Nd International Conference on Intelligent User Interfaces*, IUI '17, pages 553–557, New York, NY, USA, 2017. ACM.

Michael Kipp. Anvil: The video annotation research tool. In *Handbook of Corpus Phonology*. Oxford University Press, Oxford, UK, 2013.

Reza Lotfian and Carlos Busso. Building naturalistic emotionally balanced speech corpus by retrieving emotional speech from existing podcast recordings. *IEEE Transactions on Affective Computing*, PP(99):1–1, 2017.

Johann Poignant, Mateusz Budnik, Hervé Bredin, et al. The CAMOMILE collaborative annotation platform for multimodal, multi-lingual and multi-media documents. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation LREC 2016, Portorož, Slovenia, May 23-28, 2016.*, 2016.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1135–1144. ACM, 2016.

Stephanie Rosenthal and Anind K. Dey. Towards maximizing the accuracy of human-labeled sensor data. In *Proceedings of the 2010 International Conference on Intelligent User Interfaces, February 7-10, 2010, Hong Kong, China*, pages 259–268, 2010.

Wojciech Samek, Thomas Wiegand, and Klaus-Robert Müller. Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models. *CoRR*, abs/1708.08296, 2017.

Burr Settles. Active learning literature survey. 52(55–66):11 pages, 2010.

Frode Sørmo, Jörg Cassens, and Agnar Aamodt. Explanation in case-based reasoning–perspectives and goals. *Artificial Intelligence Review*, 24(2):109–143, Oct 2005.

Alessandro Vinciarelli, Maja Pantic, and Hervé Bourlard. Social signal processing: Survey of an emerging domain. *Image Vision Computing*, 27(12):1743–1759, November 2009.

Johannes Wagner, Elisabeth André, Michael Kugler, and Daniel Leberle. SSI/ModelUI - A tool for the acquisition and annotation of human generated signals. In *DAGA 2010*, Berlin, Germany, 2010. TU Berlin.

Johannes Wagner, Florian Lingenfelser, Tobias Baur, Ionut Damian, Felix Kistler, and Elisabeth André. The social signal interpretation (ssi) framework: multimodal signal processing and recognition in real-time. In *Proceedings of the 21st ACM international conference on Multimedia*, MM '13, pages 831–834, New York, NY, USA, 2013. ACM.

Johannes Wagner, Tobias Baur, Yue Zhang, Michel F. Valstar, Björn W. Schuller, and Elisabeth André. Applying cooperative machine learning to speed up the annotation of social signals in large multi-modal corpora. *CoRR*, abs/1802.02565, 2018.

Peter Wittenburg, Hennie Brugman, Albert Russel, Alex Klassmann, and Han Sloetjes. Elan: A professional framework for multimodality research. *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC)*, pages 879–896, 2006.

Huijun Wu, Chen Wang, Jie Yin, Kai Lu, and Liming Zhu. Sharing deep neural network models with interpretation. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*, pages 177–186. International World Wide Web Conferences Steering Committee, 2018.

Yue Zhang, Eduardo Coutinho, Björn Schuller, Zixing Zhang, and Michael Adam. On rater reliability and agreement based dynamic active learning. In *International Conference on Affective Computing and Intelligent Interaction, ACII*, pages 70–76, Xi'an, China, September 2015.

Yue Zhang, Eduardo Coutinho, Zixing Zhang, Caijiao Quan, and Björn Schuller. Agreement-based dynamic active learning with least and medium certainty query strategy. In A Krishnamurthy, A Ramdas, N Balcan, and A Singh, editors, *Proceedings of the 32nd International Conference on Machine Learning (ICML 2015). JMLR W&CP volume 37*, pages 1–5, Lille, France, 2015. Lille, France.

Zixing Zhang, Eduardo Coutinho, Jun Deng, and Björn Schuller. Cooperative learning and its application to emotion recognition from speech. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(1):115–126, Jan 2015.

Xiaojin Zhu. Semi-supervised learning literature survey. Technical report, Computer Sciences, University of Wisconsin-Madison, 2005.

Learning Attributions Grounded in Existing Facts For Robust Visual Explanation

Yulong Wang, Xiaolin Hu, Hang Su
Department of Computer Science and Technology, Tsinghua Lab of Brain and Intelligence
State Key Lab for Intell. Tech & Sys., BNRist Lab, Tsinghua University, 100084, China
wang-yl15@mails.tsinghua.edu.cn; {xlhu, suhangss}@mail.tsinghua.edu.cn

Abstract

Visual explanation, which aims to interpret the model's output by attributing to input features, functions as an intuitive form for interpreting black-box models. However, previous works over-emphasize foreground saliency visualization, which either generate similar saliency maps regardless of the class requested for explanation, or generate dispersive artifacts when requested class does not exist in the input image. We present the reasons for these flaws as mode collapse and adversarial noise, which are previously observed in the research of generative models. To overcome these shortcomings, we propose a general principle for perturbation-based optimization by learning attributions grounded in *existing* facts for robust visual explanation. We further propose a Hierarchical Attribution Fusion (HAF) technique to mitigate the artifacts without extra smoothing regularization. To evaluate the visual explanation more thoroughly, a new evaluation task Distracted Weakly Supervised Object Localization (D-WSOL) is proposed to measure whether these methods can correctly attribute the output of requested class to the existing facts. Experiments show that previous methods fail to conform to this criteria, and our principle can help improve the robustness by suppressing false saliency response.

1 Introduction

Deep convolutional neural networks have achieved superior performance in recent years on various vision tasks, including object recognition [He *et al.*, 2016; Huang *et al.*, 2017], detection [Ren *et al.*, 2015], segmentation [He *et al.*, 2017], etc. The powerful representation learning ability of CNNs has also been validated beyond the above specific fields, like in deep reinforcement learning [Silver *et al.*, 2017], neural machine translation [Wu *et al.*, 2016], etc. However, with the increasing utilization of neural network models in daily life services, the interpretability of networks has become a demanding request for safety concern, especially in critical industry like medical diagnosis [Zhang *et al.*, 2017], self-driving cars [Kim and Canny, 2017].

To interpret neural networks, especially CNNs with visual inputs, one can apply *visual explanation* by attributing the model's prediction to the input features, and representing saliency in the visual form [Simonyan *et al.*, 2013; Springenberg *et al.*, 2014; Zhou *et al.*, 2016]. The attribution results can help identify whether the model's output conforms to human common knowledge. Also, in some label-scarce situation, the visual saliency can provide pixel level annotation, which can be further supporting facts for the validation of model's decision [Baumgartner *et al.*, 2017].

Previous approaches mainly focus on the final visual saliency on input features and accurate saliency generation, which diverts from the fundamental problem that explains the significance of input features *actually* contributing to the final output computation. A more severe problem is when the class requested for explanation does not exist in the input image, these methods either generate similar saliency maps or dispersive artifacts, in contrast to suppressing saliency response as a robust explainer conforming to human perception should perform. The false saliency response can be particularly dangerous in the visualization-based disease diagnosis. These flaws reflect that current methods fail to perform robust explanation based on the existing facts.

In this paper, we propose a general principle for perturbation-based visual explanation methods by learning attributions grounded in *existing* facts. The core idea is to keep the model's output close to the original value after occluded by the saliency mask. This avoids adversarial noise generation, which can be caused by blindly increasing requested class score without considering the existing facts. We also propose a Hierarchical Attribution Fusion (**HAF**) technique for mitigating artifacts by combining network's intermediate attributions, without introducing extra regularization terms. These hierarchical attributions can also help reveal the network's intermediate attentions on different layers.

A common evaluation for visual saliency is Weakly Supervised Object Localization (WSOL) by predicting object bounding box given ground-truth label. Other saliency metrics [Dabkowski and Gal, 2017] and tasks [Fong and Vedaldi, 2017] are also proposed to overcome the limitation of localization-based metric. However, these tasks overemphasize the performance of explaining ground-truth class, failing to reflect whether the method attributes requested class to input features, or just performs saliency detection. There-

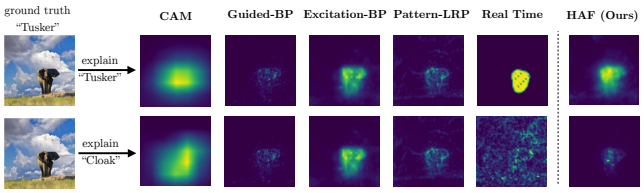


Figure 1: Comparison of several recent visual explanation approaches. When explaining the ground-truth class existing in the input image, all methods can localize the object correctly, as shown in the first row. However, when explaining a non-existing class, these approaches either still generate similar saliency map regardless of requested class or generate dispersive and irregular artifacts. On the contrary, our method has suppressed most of the responses compared to other methods.

fore we propose a new evaluation task termed as Distracted Weakly Supervised Object Localization (**D-WSOL**), to overcome the sole emphasis on localizing ground-truth objects. The new task considers the capability of suppressing saliency response when requesting non-existing class, while keeping the performance of localizing ground-truth class. Experiments show that previous methods fail to conform to the new criteria, and the proposed principle can help improve the visual explanation robustness.

2 Related Work

Recent visual explanation approaches can be categorized into *gradient-based* methods and *perturbation-based* methods [Ancona *et al.*, 2018]. Gradient-based methods conduct visual attribution progress starting from the output layer by following predefined backpropagation rules [Simonyan *et al.*, 2013; Springenberg *et al.*, 2014; Zhang *et al.*, 2016; Montavon *et al.*, 2017]. [Ancona *et al.*, 2018] summarizes several of these methods in a general framework. Perturbation-based methods generate visual saliency by comparing the output changes when adding perturbations onto parts of input features. A naive approach is to occlude parts of input extensively and collect output changes as the indication of saliency [Zeiler and Fergus, 2014]. More efficient methods have been proposed to accelerate this progress [Zintgraf *et al.*, 2017] and reduce artifacts by adding extra regularizations [Dabkowski and Gal, 2017; Fong and Vedaldi, 2017].

There are also different approaches for visual explanation besides the above two types. Class Activation Map (CAM) [Zhou *et al.*, 2016] and its variant Grad-CAM [Selvaraju *et al.*, 2017] visualize class-specific saliency based on the correlation between high layer features and class representative pattern weights. Net-dissection [Bau *et al.*, 2017] can localize the high level semantics for the corresponding filters. More comprehensive reviews can be referred to [Guidotti *et al.*, 2018; Zhang and Zhu, 2018].

3 Flaws in current visual explanation methods

Before introducing our proposed method, we first discuss the two flaws of current visual explanation methods. Figure 1 displays visual attribution results including CAM [Zhou *et al.*, 2016], Guided-Backpropagation [Springenberg *et al.*, 2014], Excitation-Backpropagation [Zhang *et al.*, 2016], Pattern-LRP [Kindermans *et al.*, 2017] and Real Time Saliency [Dabkowski and Gal, 2017]. When explaining the ground-truth class existing in the input image, all methods can localize the input features correctly, though there are differences in the extent and intensity. However, these approaches will still generate high-response output even if non-existing class is requested for explanations, which a robust explainer conforming to human perception should *not* perform. Here we present the explanations for the failure for these types of methods.

3.1 Mode collapse for gradient-based methods

The phenomenon that explainer always generates similar saliency map regardless of the requested class can be interpreted as **mode collapse** [Che *et al.*, 2016; Salimans *et al.*, 2016], in which GAN only characterizes a few mode when generator is too complex without regularization. For most gradient-based methods, the attribution process is similar to standard backpropagation, except that the attribution values are propagated from top layer to bottom layer with the modified chain rules [Ancona *et al.*, 2018]. From this perspective, the visual explanation is actually the output of a new generator network, which is *inverted* based on the original model structure and receives class indicator vector as input. Therefore when explaining non-existing class or changing the class indicator vector, the generator network without fully trained and regularized will fall into the single mode, only generating similar salient map.

3.2 Adversarial noise for perturbation-based methods

For perturbation-based methods, the phenomenon that explainer generates irregular and dispersive saliency map when explaining non-existing class can be interpreted as **adversarial noise** [Szegedy *et al.*, 2013; Goodfellow *et al.*, 2014] superposed on the input features, which is caused by blindly optimizing the saliency mask towards a biased target class. Specifically, the visual saliency is obtained by optimizing mask over input features with certain objective, which can be generally formulated as

$$\min_{M} \; -\log f_c(\Phi(M, I)) + \Omega(M), \tag{1}$$

where M is saliency mask, f_c is softmax probability for the class cr, Φ is operation of removing parts of input image I by M, and Ω is regularization function to constrain M to have certain prior property, such as smoothness and sparsity.

The objective is to encourage the masked input to activate the corresponding class probability as high as possible under the constraint Ω. However, when requested class c is a non-existing class in the input features, only pursuing high probability saliency mask will result in adversarial noise superposed on the input. Actually, when assuming $\Phi(M, I) = M \odot I$ where $\odot$ is element-wise multiplication, the adversarial noise ΔI added onto the input image is parametrized by

$$\Delta I = (M_{c_n} - 1) \odot I, \tag{2}$$

where M_{c_n} is fully optimized for target class c_n.

4 Learning Hierarchical Attributions Grounded in Existing Facts

The major flaw of previous visual explanation methods is that the explanation is not grounded in *existing* facts, and the resulting visual saliency is biased towards an idealized categorical output. To avoid adversarial noise generation during attribution process, we propose a basic requirement for a robust visual explanation that it should explain the actual output for any given requested class.

4.1 Explanation grounded in existing facts

Specifically, for perturbation-based methods, the objective in Equation (1) should be modified as

$$\min_{M} \ ||s_c(M \odot I) - s_c(I)||^2 + \Omega(M), \tag{3}$$

where $s_c(\cdot)$ means the pre-softmax output score for the requested class c. This objective simply requires that after input image I is perturbed by saliency mask M, the new class score should remain similar, rather than be blindly increased without considering the original value intensity.

4.2 Learning hierarchical attributions

Directly optimizing the saliency mask on input can suffer from artifacts. Previous approaches [Fong and Vedaldi, 2017; Dabkowski and Gal, 2017] adopt more regularization items or training a new scaffolding network for mask generation. Here, we propose to learn a set of hierarchical attributions to mitigate the artifacts with fewer optimization iterations and tuning hyperparameters.

Specifically, for a pretrained network Θ, the output feature maps of intermediate layers are $\mathcal{F}_1, \mathcal{F}_2, \cdots, \mathcal{F}_K$. We associate a 3D saliency mask $\mathcal{M}_i$ for each $\mathcal{F}_i$ with equal size. During forward stage, these masks will be multiplied on feature maps, which function like control gates on the intermediate outputs. These masks will be optimized together with the objective

$$\min_{\mathcal{M}_1,\cdots,\mathcal{M}_K} ||s_c(I; \mathcal{M}_1, \cdots, \mathcal{M}_K) - s_c(I)||^2 + \Omega(\mathcal{M}_1, \cdots, \mathcal{M}_K). \tag{4}$$

The final visual saliency mask is the fusion of these hierarchical attributions by

$$M = \frac{1}{K}\sum_{i=1}^{K} \frac{1}{Z_i} \text{resize}(\text{channel\_mean}(\mathcal{M}_i)), \tag{5}$$

where channel_mean takes the mean of 3D attribution across channels into 2D visual saliency map, resize upsamples high level attributions to the full image size, and Z_i is the normalization term to transform the saliency map into probability. The final mask is the aggregation of the different levels of attentions, and will not be further optimized.

The above technique named as Hierarchical Attribution Fusion (HAF) can also obtain the intermediate layers' attentions to help understand network functional behaviors. Figure 2 displays the different levels of attributions in GoogleNet, VGG16 and ResNet-50. The low level attributions tend to be dispersive and focus on image details, while high level attributions tend to be smooth and focus on the main body of the object. The fusion mask by combining hierarchical attributions can reach a balance on the intensity and extent of saliency. ResNet-50 tends to generate the most precise saliency map among these three models, which is consistent with its powerful and efficient representation learning ability.

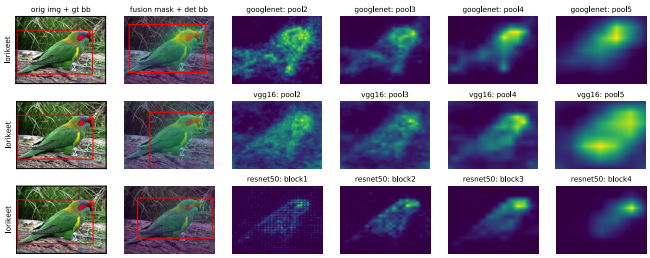


Figure 2: Different level attributions in GoogleNet, VGG16 and ResNet-50. The first column shows ground-truth images and corresponding labels and bounding boxes. The second column shows the attribution fusion results in saliency map and localization boxes by thresholding. The rest columns show attribution masks at each intermediate levels.

4.3 Implementation details

For the specific regularization form in Equation (3), we observe that the simple ℓ_1-norm is enough for learning meaningful attributions. To balance the contribution of regularization term in the total loss, we rescale the ℓ_1-norm of each layer attribution $\mathcal{M}_i$ by λt_i, where λ is a common balance factor and t_i is an adjusted factor for $\mathcal{M}_i$ based on its spatial dimension.

To avoid learning redundant attributions, we select a subset of layers as reference for explaining. We apply our approach on VGG16 [Simonyan and Zisserman, 2014], GoogleNet [Szegedy *et al.*, 2015] and ResNet-50 [He *et al.*, 2016] pretrained models. For VGG16 and GoogleNet, we select the output of four max pooling layers for learning attributions. For ResNet-50, we select the output at the end of four residual blocks for learning attributions.

5 Experiments

5.1 Visual explanation comparison

We compare our method against some representative visual explanation methods in the visual form, including guided-backpropagation (Guid), excitation-backpropagation (Exc), Pattern-LRP (Pt-LRP), Real Time Saliency (Real Time). All these results are based on the public release implementations. Figure 3 displays the comparison results. For each image with green box, we denote the situation where the requested class equals the ground-truth class (shown at the bottom), and the corresponding row shows the image overlaid by final saliency map. For each image with red box, we denote the situation where requested class is a non-existing class in the input (shown at the bottom), and the corresponding row shows the visual saliency only. At the bottom of each saliency map, we also display the variance of saliency map intensity after normalizing to range [0, 1]. From the figure, we can conclude that when explaining ground-truth class, Excitation-BP, Real Time Saliency, and our proposed HAF performs well, even

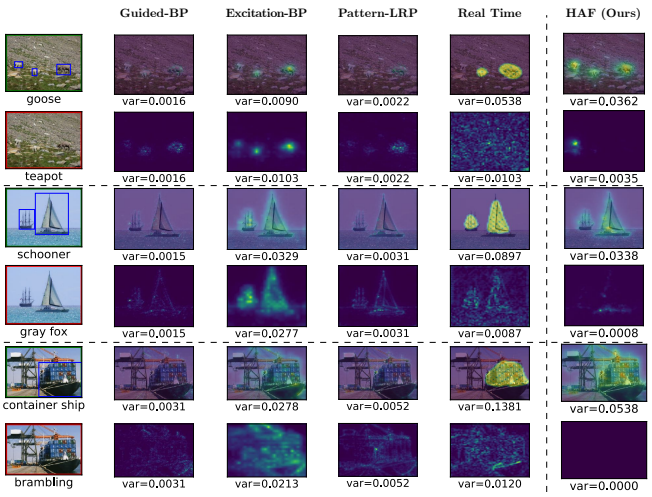


Figure 3: Visual comparison of several visual explanation methods. For each image with green box, the requested class equals the ground-truth class (shown at the bottom), and the corresponding row shows the image overlaid by final saliency map. For each image with red box, the requested class is a non-existing class in the input (shown at the bottom), and the corresponding row shows the visual saliency only. At the bottom of each saliency map, we also display the variance after normalizing to range [0, 1], which should be close to zero for non-existing requested class.

in the co-occurrence of multiple objects. However, when explaining non-existing class, the gradient-based methods still generate similar saliency map with that of ground-truth class. For Real Time Saliency method, it will generate dispersive saliency map. Our method can successfully suppress high response in the saliency generation with low variance.

5.2 Distracted weakly supervised object localization

Here we present a new evaluation task Distracted Weakly Supervised Object Localization (D-WSOL), which is similar to the usual Weakly Supervised Object Localization (WSOL) evaluation protocol, except that we consider the effects of non-existing requested class.

For WSOL, we adopt mean thresholding strategy to predict a bounding box. Specifically, we can threshold the saliency map $\mathcal{S}$ by $\tau = \alpha\mu_I$, where μ_I is the mean intensity of saliency map . Then the tightest bounding box that contains the whole thresholded saliency map is set as the localization box. Localization error is calculated as the IOU greater than 0.5 with any of ground-truth bounding boxes.

For D-WSOL, the saliency map used for localization is changed to

$$\mathcal{S} = \text{abs}(\mathcal{S}_{c_g} - (\mathcal{S}_{c_n} - \text{mean}(\mathcal{S}_{c_n}))), \tag{6}$$

where $\mathcal{S}_{c_g}$ and $\mathcal{S}_{c_n}$ are saliency maps generated given ground-truth class and non-existing class respectively. If the visual explanation method can suppress saliency response given non-existing class, $\mathcal{S}_{c_n}$ will have low variance and $\mathcal{S}_{c_n} - \text{mean}(\mathcal{S}_{c_n})$ should be close to zeros.

Table 1 summarizes the localization error for each methods. Our method can achieve comparable localization performance given ground-truth class. However, in the situation of D-WSOL, most of these methods result in severe performance degradation. Our method can effectively resist the disturbance caused by non-existing class and achieve the lowest localization error, indicating the improvement of visual explanation robustness.

Table 1: Weakly supervised localization errors on ImageNet validation dataset for several visual explanation methods under normal mode and distracted mode. α^* is optimized on held-out 5,000 images from ImageNet training dataset. Our method can achieve comparable localization performance in normal mode and effectively resist disturbance from non-existing class query.

Model	Method	Normal		Distracted	
		α^*	Err (%)	α^*	Err (%)
GoogleNet	Grad	5.00	41.70	2.00	46.58
	Guid	4.50	42.00	0.50	58.56
	CAM	1.00	48.10	1.00	49.08
	Exc	1.50	39.00	0.50	58.57
	Feed	–	**38.80**	2.00	47.14
	Ours	1.50	41.30	1.50	**41.98**
VGG16	Pt-LRP	3.50	47.60	0.50	58.57
	Ours	1.50	**40.80**	1.00	**43.78**
ResNet-50	Real Time	–	**36.70**	0.50	42.67
	Ours	1.50	39.20	1.50	**39.65**

6 Conclusion

In this paper, we discuss the flaws of current visual explanation methods, which include always generating similar saliency maps regardless of requested class and generating dispersive visual saliency when given non-existing class. These flaws can be attributed to mode collapse and adversarial noise. The main reason is that current approaches fail to conform the principle of explaining existing facts in input features. Therefore, we propose a general optimization objective by keeping the new output score close to the original value while learning attributions. We also propose Hierarchical Attribution Fusion technique to reduce artifacts without extra regularizations. To evaluate the robustness of visual explanation methods, a new evaluation task termed as Distracted Weakly Supervised Localization is proposed to evaluate both the performance of localizing ground-truth class and capability of suppressing disturbance from non-existing class. Both visual assessment and evaluation metrics validate that our method can generate more robust explanations. Future works include incorporating the proposed method into visual detection system to help discover adversarial samples.

References

[Ancona *et al.*, 2018] Marco Ancona, Enea Ceolini, Cengiz Oztireli, and Markus Gross. Towards better understanding of gradient-based attribution methods for deep neural networks. In *ICLR*, 2018.

[Bau *et al.*, 2017] D. Bau, B. Zhou, A. Khosla, A. Oliva, and A. Torralba. Network dissection: Quantifying interpretability of deep visual representations. In *CVPR*, 2017.

[Baumgartner *et al.*, 2017] Christian F Baumgartner, Lisa M Koch, Kerem Can Tezcan, Jia Xi Ang, and Ender Konukoglu. Visual feature attribution using wasserstein gans. *arXiv preprint arXiv:1711.08998*, 2017.

[Che *et al.*, 2016] Tong Che, Yanran Li, Athul Paul Jacob, Yoshua Bengio, and Wenjie Li. Mode regularized generative adversarial networks. *arXiv preprint arXiv:1612.02136*, 2016.

[Dabkowski and Gal, 2017] Piotr Dabkowski and Yarin Gal. Real time image saliency for black box classifiers. In *NIPS*, 2017.

[Fong and Vedaldi, 2017] Ruth C Fong and Andrea Vedaldi. Interpretable explanations of black boxes by meaningful perturbation. In *CVPR*, 2017.

[Goodfellow *et al.*, 2014] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv:1412.6572*, 2014.

[Guidotti *et al.*, 2018] Riccardo Guidotti, Anna Monreale, Franco Turini, Dino Pedreschi, and Fosca Giannotti. A survey of methods for explaining black box models. *arXiv preprint arXiv:1802.01933*, 2018.

[He *et al.*, 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[He *et al.*, 2017] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, 2017.

[Huang *et al.*, 2017] Gao Huang, Zhuang Liu, Kilian Q Weinberger, and Laurens van der Maaten. Densely connected convolutional networks. In *CVPR*, 2017.

[Kim and Canny, 2017] Jinkyu Kim and John Canny. Interpretable learning for self-driving cars by visualizing causal attention. In *CVPR*, 2017.

[Kindermans *et al.*, 2017] Pieter-Jan Kindermans, Kristof T Schütt, Maximilian Alber, Klaus-Robert Müller, and Sven Dähne. Patternnet and patternlrp–improving the interpretability of neural networks. *arXiv preprint arXiv:1705.05598*, 2017.

[Montavon *et al.*, 2017] Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. Methods for interpreting and understanding deep neural networks. *Digital Signal Processing*, 2017.

[Ren *et al.*, 2015] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*, 2015.

[Salimans *et al.*, 2016] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *NIPS*, 2016.

[Selvaraju *et al.*, 2017] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *ICCV*, 2017.

[Silver *et al.*, 2017] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 2017.

[Simonyan and Zisserman, 2014] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[Simonyan *et al.*, 2013] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv:1312.6034*, 2013.

[Springenberg *et al.*, 2014] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.

[Szegedy *et al.*, 2013] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv:1312.6199*, 2013.

[Szegedy *et al.*, 2015] C Szegedy, W Liu, Y Jia, P Sermanet, S Reed, and D Anguelov. Going deeper with convolutions. In *CVPR*, 2015.

[Wu *et al.*, 2016] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv:1609.08144*, 2016.

[Zeiler and Fergus, 2014] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *ECCV*, pages 818–833, 2014.

[Zhang and Zhu, 2018] Quan-shi Zhang and Song-Chun Zhu. Visual interpretability for deep learning: a survey. *Frontiers of Information Technology & Electronic Engineering*, 2018.

[Zhang *et al.*, 2016] Jianming Zhang, Zhe Lin, Jonathan Brandt, Xiaohui Shen, and Stan Sclaroff. Top-down neural attention by excitation backprop. In *European Conference on Computer Vision*, pages 543–559. Springer, 2016.

[Zhang *et al.*, 2017] Zizhao Zhang, Yuanpu Xie, Fuyong Xing, Mason McGough, and Lin Yang. Mdnet: A semantically and visually interpretable medical image diagnosis network. In *CVPR*, 2017.

[Zhou *et al.*, 2016] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *CVPR*, 2016.

[Zintgraf *et al.*, 2017] Luisa M Zintgraf, Taco S Cohen, Tameem Adel, and Max Welling. Visualizing deep neural network decisions: Prediction difference analysis. *arXiv preprint arXiv:1702.04595*, 2017.

Explicating feature contribution using Random Forest proximity distances

Leanne S. Whitmore, Anthe George, Corey M. Hudson
Sandia National Laboratories, Livermore, CA 94551
cmhudso@sandia.gov

Abstract

In Random Forests, proximity distances are a metric representation of data into decision space. By observing how changes in input map to the movement of instances in this space we are able to determine the independent contribution of each feature to the decision-making process. For binary feature vectors, this process is fully specified. As these changes in input move particular instances nearer to the in-group or out-group, the independent contribution of each feature can be uncovered. Using this technique, we are able to calculate the contribution of each feature in determining how black-box decisions were made. This allows explication of the decision-making process, audit of the classifier, and *post-hoc* analysis of errors in classification.

1 Introduction

Limitations in the ability to identify the pathway to decision-making for individual instances are the fundamental justification for a classifier being labeled "black box" [Caruana *et al.*, 1999]. Visibility of the decision-making for individual cases through a machine learned model, what we here refer to as *explication*, is essential for algorithmic transparency [Lipton, 2016]. One of the most popular "black box" classifiers is the Random Forest Classifier and its associated variants (including Bagging Classifiers and Gradient Boosted Trees Classifiers). Despite the common use of these techniques for machine learning, in high risk or high consequence settings, lower accuracy methods are often preferred, due to their interpretability [Caruana *et al.*, 2015].

The manner in which Random Forests work has been well described by Breiman (2001), including how they limit overfitting (using the Strong Law of Large Numbers) and achieve their level of accuracy (through improvements to bagging and boosting algorithms and decorrelation of features in random samples). In spite of Random Forests implementing a straightforward construction algorithm, the intricacies of a model's decision-making processes is largely hidden from the outcome [Friedman *et al.*, 2001]. This separates Random Forests from techniques like Logistic Regression Classification, in which individual coefficients correspond directly to weighting factors [Cooper *et al.*, 1997]. With these more interpretable, but simpler models, running an individual instance through the model, is useful in explaining the decision-making process for that instance. The coefficients in a logistic regression model correspond directly to the odds outcome weightings for each feature. Decisions, therefore can be made, using these features that will directly affect the outcome (e.g., [Herrmann *et al.*, 1992]). However, logistic regression is seldom very accurate and in high risk setting, operators are frequently forced to choose between less accurate methods with more explainable models [Bratko, 1997; Caruana, 2017].

1.1 Model feature importances are limited in terms of explicability

Most of the interpretability in Random Forests come from a set of outcome variables, referred to as feature or variable importances. These values are calculated a number of ways, but are generally measured as the mean decrease in accuracy across the model after traversing tree nodes and splitting on the feature of interest [Friedman *et al.*, 2001]. As Breiman and Cutler (2008) describe it, this requires a simple random permutation of a variable in the sampled instances, run through the model. The number of correctly classified counts are then compared between the sampled instances (without permutation) and the sampled instances (with permutation). The average over all trees is the feature importance score for that variable.

One of the limitations of feature importances, in terms of interpretability, is that they measure the importance of the feature in the model, and not to individual cases. The manner in which feature importances relate to individual instances is not clear from the feature importance values. Highly weighted features may have little importance for a particular instance. They may also be positive indicators of class membership for one instance and negative indicators of class membership for another instance, even when the outcome prediction is identical. Feature importances give global importance, but not local importance or the direction in which they drive a particular decision-making process. The global role of features importance measurements cannot provide the level of explanation that is frequently found in more causal measures, like interventions [Garant and Jensen, 2016; Galles and Pearl, 1995].

1.2 Related work in feature contribution and explicability

Recently, a variety of techniques, under the general banner of "counterfactual probing" techniques have emerged in machine learning [Voosen, 2017; Wachter *et al.*, 2017]. These techniques attempt to explain and explicate machine learning decision-making processes. In general, the central idea of these techniques, has been to insert randomness into the feature space, in order to determine the resulting outcome in decision space [Lundberg and Lee, 2017b; Štrumbelj and Kononenko, 2014]. One of the most general purpose of these is LIME (Locally Interpretable Machine-Agnostic Explanations), which maps a complicated and uninterpretable model into linear space, to provide local model fidelity and explainability [Ribeiro *et al.*, 2016]. Other work in this area has involved determining output or internal neural network gradients through input perturbations [Shrikumar *et al.*, 2017; Lengerich *et al.*, 2017], additive feature coalition detection [Lundberg and Lee, 2017a], and probability series expansion of feature groupings [Agarwal and Hudson, 2017].

The primary contribution of this paper is to provide work parallel to the above-listed works on explainability/explicability. Rather than creating a new classifier or new metrics for interpreting classifiers, we seek to provide a topological interpretation of proximity (a well known metric) for Random Forests (a commonly used classifier). Using this topological interpretation, we are able to show how the perceived black-box outcome of individual Random Forests decision-making processes can be made explicable through random permutation of given feature values.

2 Representing Random Forests in Decision Space

By default, many implementations of Random Forest classification use the CART algorithm. This algorithm represents the decision-making process as a set of discrete splits, ultimately creating binary decision trees [Breiman *et al.*, 1984]. Within the mapping from feature vectors to each decision tree, the data is re-represented as a discrete binary vector. Determining the contribution of individual features in the Random Forests decision-making process, requires determining how each value in feature space maps an instance in decision space. For classification operations on binary feature vectors, this process for independently described features is fully specifiable - meaning that the full range of variability in binary features can be captured in the model. In many classification and regression tasks, specification of only binary features is an unsuitable abstraction. However, the underlying data structure of Random Forests is the CART decision tree. To accommodate this data structure, categorical and continuous variable are discretized into binary features. For example the single feature $X_i \in a, b, c$ is frequently converted into three binary features $X_{ia} \in \{0, 1\}$, $X_{ib} \in \{0, 1\}$, $X_{ic} \in \{0, 1\}$ and $X_i = \{x \in \mathbf{R} \mid 0 \leq x \leq 1\}$ is frequently split into discrete parts, determined by the optimal Gini impurity, e.g., $X_{i1} = \{x \in \{0, 1\} \mid x < 0.25\}$, $X_{i2} = \{x \in \{0, 1\} \mid x < 0.71\}$, $X_{i3} = \{x \in \{0, 1\} \mid x > 0.5\}$, in which case a value x = 0.3 will correspond to a feature vector 010 and a value x = 0.65 will correspond to a feature vector 011. This is a fundamental characteristic of tree-based classifiers, and separate this class of machine learning methods from methods that use a maximally discriminative hyperplane, like Support Vector Machines or Artificial Neural Networks. Random Forests make decisions about how to convert numerical features to binary features. Between user input and the creation of binary feature vectors there is a process of choosing how to split numerical variables. One reason to specifically focus on binary feature vectors is to address feature value permutation, rather than the process of turning numerical variables into binary variables. A numerical feature may have values between 0 and 100, but split at 1.1. Changing feature values across the range of values is meaningless if it undersamples between 0 and 1.1 and oversamples between 1.1 and 100. If only two binary features correspond to the numerical value then there are only two feature values that contribute to the predicted outcome.

2.1 Defining a proximity distance

The determination of proximity between two instances in a Random Forest data set was defined by Breiman (2001) as the number of times two instances share a leaf node in a Decision Tree divided by the number of trees in a Random Forest. Proximity is a similarity metric that accounts for the similarity in the outcome of a hidden decision-making process. Since proximity is a dot-product between 0/1 vectors, per the leaf indicator functions, the Euclidean space that Breiman (2001) defines is necessarily similar to the Hamming distance, which has previously been shown to satisfy the properties of a metric.

2.2 Proximity distance defines the Random Forest decision space

The totality of all $D(x, .)$ for all x in the training set is the training proximity distance space P. All instances sharing the same class as a particular instance x can be labeled as an instance's in-group. $P_x(in)$ defines the covering of in-group proximity space. $P_x(out)$ covers all other distances, and can be labeled as the out-group proximity space for instance x. Breiman (2002) defined the average proximity for case n to all other training instances k, where $n \neq k$ in class j as:

$$\bar{P} = \sum_{n=1}^{\|n \in (class(j))\|} proximity(n, k)^2 \qquad (1)$$

This value can be defined both for the in-groups and out-groups. This can be similarly be defined for proximity distance (1 - proximity) with the opposite interpretation. In this case, small average in-group proximity distances correspond to closeness in decision space. Breiman and Cutler (2001) point out that this measure can also be used to threshold outliers, for instances where the average in-group proximity distance is large.

The complete matrix of proximity distances defines the proximity space. Distances in decision space, may be quite different than distances in feature space. In feature space, the feature vectors are weighted, either equally, or by *a priori* determination. The importance of a value in the input vector, with regard to discriminating the instance into a class

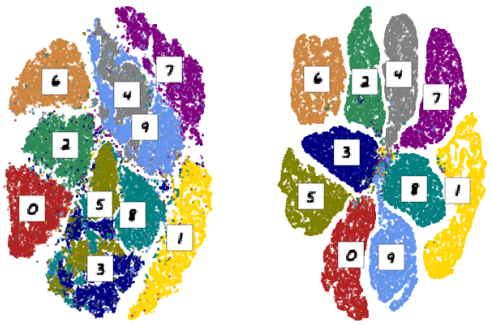


Figure 1: Data are plotted using t-SNE (perplexity = 30, number of iterations = 1000, learning rate = 200). For the distance space figure (*left*) the data are Hamming distances of binary feature vectors. For the proximity distance space (*right*) the data are transformed using proximity and then mapped into lower dimensional space with t-SNE. Kullback-Leibler divergence for the Hamming distance projection is 3.21 and for the proximity distance projection is 2.96.

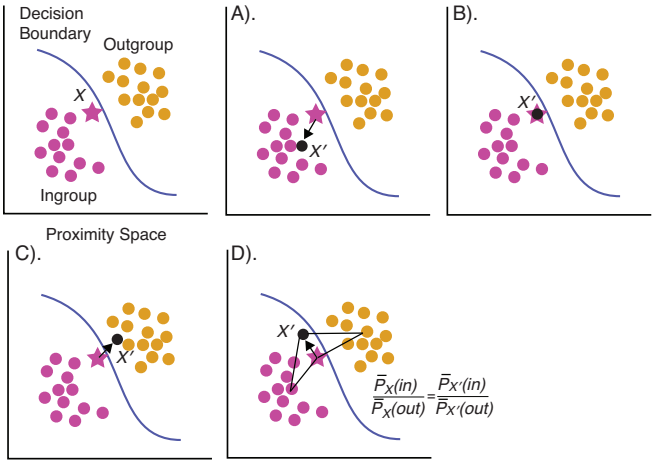


Figure 2: Results of k value permutation. The space (P) is defined by all relative distances. The decision boundary here is a simple function. In real decision space, for Random Forests, these are often high-dimensional, discontinuous and non-linear. Relative to X input vector, there is an ingroup and an outgroup. A) A change in k, may move X' closer to the ingroup. This suggests that k weakened the ultimate decision, proportional to the distance between X and X'. B) A change in k, may have little change in the relative position of X' with regard to X, suggesting that k had little to no influence on the decision. C) A change in k may result in X' being closer to the outgroup. This suggests that k contributed to the ultimate decision, at a strength proportional to the distance between X and X'. D) Suggests that a change in k resulted in X' being proportionally equivalent to both the ingroup and outgroup. If $\bar{P}_x(in) < \bar{P}_{x'}(in)$, then k contributed to X being an inlier in decision space. If $\bar{P}_x(in) > \bar{P}_{x'}(in)$, then k contributed to X being an outlier in decision space.

is therefore unknown, although, this is frequently a goal of unsupervised learning. Proximity distances transform input features through the decision trees in the Random Forest (see Fig. 1). As such, sharing few highly discriminating features may be more important for placing two instances in a leaf node than sharing many more features with less discriminating power. In many ways, this is a goal in feature salience studies, which seek to reduce the space of features [Wang *et al.*, 2001] - however, our technique uses all provided features, in order to create a proximity space.

2.3 Using Proximity Distance to explicate feature contribution

Zhou et al. (2010) recognized that change in proximity matrices could be used to identify model feature importances. However, their study of genomic features compared the result of different treatments in the decision-making process. In this work, we seek to individually explain the manner in which features independently contribute to the specific decision-making processes. This is the fundamental difference between feature importance (a property of the model) and feature contribution (a property of the individual decision-making process). As mentioned above, when the data are presented in binary feature vectors, the process of identifying the contribution of specific features in the decision making for individual instances is fully specifiable.

Each value in an input feature vector has the potential to contribute to the output decision for a particular instance, by driving the decision-making process toward a different leaf node in the decision trees. Changing a value in a feature vector and running them through the decision trees in a Random Forest can have the effect of moving an instance nearer or farther from the in-group and out-group, orthogonal to both, or resulting in no change in P. A given feature, k, in a binary feature vector v, maps to $\varphi(v)$. Bit-wise permutation of the value k, in v, defined as v', maps to $\varphi(v')$. Distance in proximity distance space, can be determined by fixing P for the original dataset and determining a new vector of distances for $D(v', .)$ for all x in P. The change in position between $D(v', .)$ and $D(v, .)$ determines the contribution of the value of feature k, given identical values for all other features, between v' and v.

The value of k in the original input feature vector may result in one of several potentially overlapping outcomes in decision space. 1) The value may strongly contribute to the decision. 2) The value may weakly impact on the decision. 3) The value may have no impact on the decision. 4) The value may result in the outcome being closer to in-group. 5) The value may result in the outcome being closer to out-groups. The input value may result in a larger average in-group proximity distance, and larger out-group proximity distance (the value is highly discriminating, but moves the instance toward becoming an outlier). The input value may result in a larger average in-group proximity distance and smaller out-group proximity distance (the value moves the instance toward other classes). The input value may also lead to no change in the placement of the instance in the leaf nodes, leading to no change, or strictly

orthogonal placement in decision space. A hypothetical set of outcomes in decision space can be visualized in Fig 2.

3 Algorithmic Determination of Feature Contribution using k-permutations

Using the logic of 2.3 as our basis, we here use permuted bit-flipping to determine the contribution of a feature in mapping instance values in decision space. This requires a binary feature vector (v), and a Random Forest classifier (C) that was trained with X. The class of (v) is c. The $ProximityDistance$ is calculated as the number of features subtracted by the number of shared terminal nodes (between the original and permuted values) divided by the number of features. Each pair of values (original vs. permuted) are then subtracted and squared, giving a distance.

3.1 Feature contribution and closeness

There are two main values for each feature in decision space: contribution and closeness. Contribution quantifies the independent influence of each feature on the final position of an instance in proximity distance space. This is determined by modifying each feature value and measuring the distance between the original (v) and modified instances (v'). The measurement of contribution is done by comparing each proximity distance between v and all v'. Absolute distance is simply calculated as $D(v, v')$. Absolute distance, however gives no sense of contribution of the feature to the decision-making process, only the relative number of changes in the word vectors $\varphi(v)$ and $\varphi(v')$.

Feature contribution can be calculated by creating an in-group vector $\mathbf{z}$ relative to the class c of v, the size of the number of instances in the training set X. In this case

$$z_i \begin{cases} -1 & \text{if } C(v) = C(X_i) \\ 1 & \text{if } C(v) \neq C(X_i) \end{cases} \quad (2)$$

Calculating the dot-product of $\mathbf{z}$ by the squared difference between the distance vectors determines the magnitude and direction of the contribution of the feature changed in v'. Although not defined in the algorithm, these may also be scaled by the number of instances in the training set.

Large positive values in the contribution suggest the original value of the feature, permuted in v', contributed strongly to v being in the decided class. Small positive values suggest the feature weakly contributed to v being in the decided class. Large negative values suggest that the original value of the feature that was permuted in v' strongly moved v away from the decided class. Small negative values suggest that the feature weakly moved v away from the decided class. Values of zero in the contribution suggest that the feature did not contribute to the decision-making process.

Implicit in the calculation of contribution is also one of closeness. Closeness is the position in decision space of v' relative to members of the in-group and out-group of v. A feature may change how close v is to other members of the in-group and out-group. This can be calculated as in Equation 1. The proximity distances, however, are between v and each v' for each member of the class (either in-group or out-group). This determines how each feature contributed to the instance v being nearer or further from the in-group or out-group.

4 Application of Feature Contribution and Closeness

Feature contribution and closeness provide a new perspective on how data is used by the classifier. As such, we will show three main uses: 1) Demonstration of the decision-making process; 2) Explication of classification error; and 3) Inspection and identification of outliers in decision space.

4.1 Classifier audit: Chemical structures for use in biofuels

Prior work on chemical activity prediction [Whitmore *et al.*, 2016a] and structural interpretation [Whitmore *et al.*, 2016b] showed how Random Forest classifiers could be used with LIME [Ribeiro *et al.*, 2016] to elucidate the contributions of molecular structural to the classification of chemicals in terms of research octane number. Assessing the potential octane performance of novel chemicals is an important roadblock in novel biofuel adoption. In addition to research octane number, cetane is used in diesel and diesel-like fuels as a measure of chemical performance. Here we use a strictly structural Random Forest model, trained on 361 compounds to classify compounds as having either a high (> 40) or low ($<= 40$) cetane number, relative to the North American standard for diesel fuels. Because cetane is principally a suitable vs. unsuitable designator, this problem is a classification problem. Additionally, the prior research octane classifier (> 94.4 vs $<= 94.4$) classification is also examined.

Using a 6135 solely structural features (i.e., based on bond and atom presence vs. absence), a simple cetane Random Forest model has 83.04% accuracy averaged over 100 permutations with 50% randomized holdout and 85.33% accuracy with 10x cross validation. This Random Forest cetane classifier allows rapid prediction and ranking of novel chemicals in terms of potential performance in diesel-like engines. The previously reported research octane classifier, performs with 85.89% accuracy over 100 permutations with 50% randomized holdout and 86.39% accuracy with 10x cross validation.

One of the challenges to adoption of machine learning in biofuels research has been demonstrating the intuition behind the decision-making. The economic motivation prefers more time on model building to reduce risk during the expensive testing and roll-out phase. In this analysis we evaluate cetane and research octane number classification of these chemicals. These fuel characteristics are at a fundamental level, conversely related - albeit measured in independent ways. Cetane measures the propensity of a chemical to ignite under compression (the reason for it being used as a quality measure for diesel engines), whereas research octane number measures a chemical's resistance to pre-ignition (the reason for it being used as a quality measure for spark ignition engines). Good intuition for the suitability of a classifier would find that the presence of structural characteristics in a molecule would contribute in opposite ways to the high cetane and high research octane number. Since the models are using the same features, but not sharing decisions, robust models should come to the

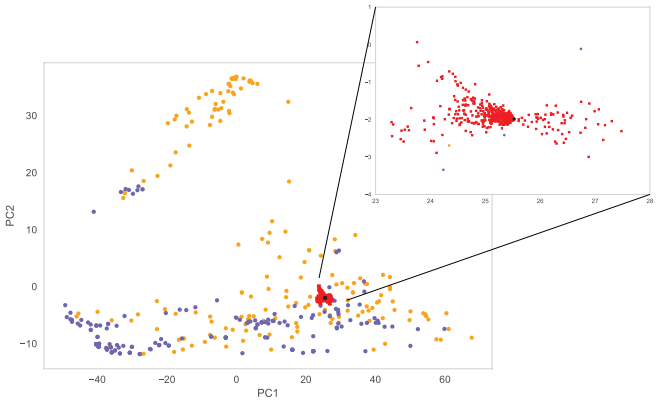


Figure 3: Low-dimensional proximity distance space (mapped onto the first two Principal Components) for 2-Methyltetrahydrofuran in cetane model. Orange dots are predicted low cetane and purple dots are predicted high cetane. The black dot is 2-Methyltetrahydrofuran. The red area corresponds to permuting individual features and the resulting change in space of the instance.

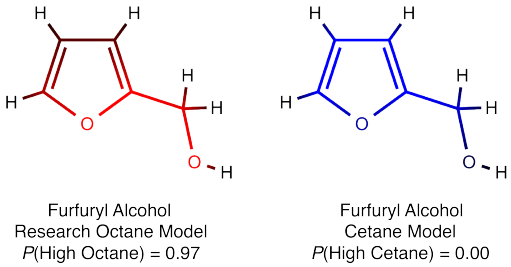


Figure 4: Mapping structural characteristics onto chemical structures. Furfuryl alcohol is a by-product of the degradation of organic products (e.g., wood, corncobs, switchgrass). This mapping varies from bright blue to bright red depending on how much each part of the molecule contributed to the high valued decision. For the research octane model (left), the bright portions contributed positively to a high octane decision. For the cetane model (right), the bright blue sections contributed negatively to a high cetane decision. The research octane model predicted that furfuryl alcohol is high octane, the cetane model predicted that it is low cetane. The mapping of individual features suggests a negative relationship between the relationships between structure and the research octane and cetane models.

same conclusions about the relationships between structure and combustion in converse directions.

Using the above proximity distance techniques it is possible to determine how changing a structural feature influences the decision-making process (see Fig 3). For instance, removing or adding a particular part of the molecule that has been classified as high cetane, may lead to the molecule moving closer or further from the other molecules that have been similarly classified.

For molecules, positive and negative contributions from in-group and out-group proximity distance for particular atoms and bonds can be remapped back onto the molecular 2-dimensional structure (see Fig 4). These maps highlight the projection of feature space, onto decision space. Each permutation in these bonds has the effect of moving the outcome nearer or further from other molecules in the in-group.

Since the underlying process in the research octane number and cetane number models is assumed to be a mapping of chemical structures to combustion characteristics, it is expected that the carry-over from the feature contributions should be jointly informative in an audit of both models. To test this, we used the entire training set for both models, 428 chemicals and determined all feature contributions for both models. In instances where at least one prediction was high (either high research octane or high cetane), all non-zero valued features were collected. The hypothesis being tested is that feature contributions between the two models should be inversely correlated, such that a negative value for a feature contribution in predicting a high value in one model (e.g., high research octane) corresponds to a positive value for the same feature in predicting the high value in the other model (e.g., high cetane).

For the 428 chemicals, there were 126,360 non-zero (i.e., $\mid x \mid > 0.0001$) features, where the chemical was predicted to be high valued in at least one of the models (this avoids the problem of cases where neither model was high valued, which may be the case for molecules that simply fail to combust).

The correlation between paired feature values between these two models is negative and significant (Pearson's $r = -0.168$, Spearman's $\rho = -0.362$, Kendell's $\tau = -0.264$, $P = 0.0$). The correlation is subject to rank issues in the dataset. Taking a simple comparison of positives vs. negatives gives similar results (see Table 1). The χ^2 statistic 14045.57 for df = 1 is significant (P = 0.0). The Pearson's Residuals are positive for (negative vs. positive) and (positive vs. negative) contingencies and negative for (positive vs. positive) and (negative vs. negative) contingencies, suggesting that the data is dominated by inverse relationships, suggesting that a feature that positively contributes to high research octane prediction, nega-

Table 1: Comparison of feature values in research octane and cetane classifiers, counts and (Pearson's Residuals)

	Research octane		
Cetane	Negative	Positive	Total
Negative	13875 (-62.43)	61634 (41.88)	75509
Positive	25338 (76.08)	25513 (-51.04)	50851
Total	39213	87147	126360

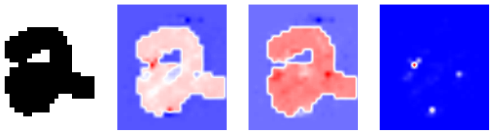


Figure 5: MNIST misclassification instance. Left panel. This is the binary representation of a two in the MNIST dataset, which was misclassified as an eight. Left center panel. This is a representation of the feature contribution for the incorrect classification. Dark red values indicate pixels that strongly weighted toward misclassification. Right center panel. This is a representation of the feature contributions for the correct classification. Dark red values indicate pixels that strongly weighted the model toward correct classification. Right panel. This is the squared difference between the two center panels. Dark red pixels correspond to differences necessary to convert between the misclassification and a correct classification.

tively contributes to high cetane prediction, and vice versa.

4.2 Classification error: Post-hoc analysis of MNIST errors

Another justification for explicable models, is understanding the failure modes. Oftentimes, even well performing models fail in ways that may be difficult to interpret or to diagnose. In this example we use 120,000 MNIST numerical figures, split between 60,000 training and six sets of 10,000 test instances. For a Random Forest, run using 1000 trees, after binary discretization of the images, the training accuracy was 100% and the testing accuracy was 97.05%. This means that 295 instances were misclassified in the testing data. Understanding why those instances were misclassified can be done using the proximity distance permutation listed above.

For a given misclassified instance, certain features contribute to the correct classification and the incorrect classification. This can be directly assessed by permuting each feature. For binary transformed MNIST data, this means bit-flipping individual pixels and determining how each one moves the instance nearer or further from the in-group.

Figure 5 illustrates an example of this. Each pixel contributes to the correct classification in a negative or positive way, by moving the instance nearer or further from other instances in the same classification. Likewise, each pixel contributes to the incorrect classification in the same manner. The difference between misclassified instance and a correctly classified instance, is that the misclassified instances has some features that overwhelm the decision-making process, pointing to the incorrect classification. The right panel in Fig 5 is a squared difference between the two feature contributions, and amounts to a permutation matrix, quantifying the transformation necessary to convert between incorrect and correct classifications.

5 Conclusion

By combining the intuition behind tools like LIME, which directly manipulate input variables and observe changes in output, with decision, rather than feature space, we have developed a tool that provides robust explication of the decision-making process. Unlike LIME, this technique is not machine agnostic, but rather specific to Random Forests. Tools like path counting and mean decrease in accuracy are notably simpler then the technique we have presented, but limit the resulting information. The proximity method has two outcome measures: 1) contribution of a feature to its prediction, and 2) closeness that a feature contributes to the in-group class membership. Both of these outcomes are important in determining how valuable a feature is in correctly or incorrectly predicting the outcome value.

Random Forests and their associated algorithms are a very commonly used set of machine learning tools. Proximity has a long history as a technique for quantifying the Random Forest decision space. We present its use in unraveling Random Forests in a way not previously described. This method is not a standard tool for sensitivity analysis in Random Forests or machine learning. The goal here is to connect a standard technique for permutation of input to the topological representation of decision space. This technique is more nuanced than changing input values and observing changes in prediction, providing users with hypothetical relationships to outcome, mapped on feature values. Three important characteristics of explainable artificial intelligence are 1) The ability to represent the individual decision-making process in human terms; 2) The ability to determine the bounds on the correctness of the algorithm, without relying simply on the measures of accuracy and error that the algorithm originally optimized; and 3) The ability to analyze decision-making errors to assess the modes of failure. Improving these three areas is an important step in developing robust, safe and responsible artificial intelligence. This work will hopefully extend intervention analysis and provide more causal explanations of data, which will lead to more useful outcomes.

Acknowledgments

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energys National Nuclear Security Administration under contract DE-NA0003525.

This research was conducted as part of the Co-Optimization of Fuels & Engines (Co-Optima) project sponsored by the U.S. Department of Energy (DOE) Office of Energy Efficiency and Renewable Energy (EERE), Bioenergy Technologies and Vehicle Technologies Offices. Co-Optima is a collaborative project of multiple national laboratories initiated to simultaneously accelerate the introduction of affordable, scalable, and sustainable biofuels and high-efficiency, low-emission vehicle engines.

References

Sapan Agarwal and Corey M Hudson. Probability series expansion classifier that is interpretable by design. *arXiv preprint arXiv:1710.10301*, 2017.

Ivan Bratko. Machine learning: Between accuracy and in-

terpretability. In *Learning, networks and statistics*, pages 163–177. Springer, 1997.

Leo Breiman and A Cutler. Random forestsclassification manual. *URL http://www. math. usu. edu/˜ adele/forests*, 2008.

Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and regression trees*. CRC press, 1984.

Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

Leo Breiman. Manual on setting up, using, and understanding random forests v3. 1. *Statistics Department University of California Berkeley, CA, USA*, 1, 2002.

Rich Caruana, Hooshang Kangarloo, JD Dionisio, Usha Sinha, and David Johnson. Case-based explanation of non-case-based learning methods. In *Proceedings of the AMIA Symposium*, page 212. American Medical Informatics Association, 1999.

Rich Caruana, Yin Lou, Johannes Gehrke, Paul Koch, Marc Sturm, and Noemie Elhadad. Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1721–1730. ACM, 2015.

Rich Caruana. Intelligible machine learning for critical applications such as health care. In *2017 AAAS Annual Meeting (February 16-20, 2017)*. aaas, 2017.

Gregory F Cooper, Constantin F Aliferis, Richard Ambrosino, John Aronis, Bruce G Buchanan, Richard Caruana, Michael J Fine, Clark Glymour, Geoffrey Gordon, Barbara H Hanusa, et al. An evaluation of machine-learning methods for predicting pneumonia mortality. *Artificial intelligence in medicine*, 9(2):107–138, 1997.

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

David Galles and Judea Pearl. Testing identifiability of causal effects. In *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*, pages 185–195. Morgan Kaufmann Publishers Inc., 1995.

Dan Garant and David Jensen. Evaluating causal models by comparing interventional distributions. *arXiv preprint arXiv:1608.04698*, 2016.

FranÇois R Herrmann, Charles Safran, Sue E Levkoff, and Kenneth L Minaker. Serum albumin level on admission as a predictor of death, length of stay, and readmission. *Archives of internal medicine*, 152(1):125–130, 1992.

Benjamin J Lengerich, Sandeep Konam, Eric P Xing, Stephanie Rosenthal, and Manuela Veloso. Towards visual explanations for convolutional neural networks via input resampling. *arXiv preprint arXiv:1707.09641*, 2017.

Zachary C Lipton. The mythos of model interpretability. *arXiv preprint arXiv:1606.03490*, 2016.

Scott M Lundberg and Su-In Lee. Consistent feature attribution for tree ensembles. *arXiv preprint arXiv:1706.06060*, 2017.

Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, pages 4768–4777, 2017.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Model-agnostic interpretability of machine learning. *arXiv preprint arXiv:1606.05386*, 2016.

Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. *arXiv preprint arXiv:1704.02685*, 2017.

Erik Štrumbelj and Igor Kononenko. Explaining prediction models and individual predictions with feature contributions. *Knowledge and information systems*, 41(3):647–665, 2014.

Paul Voosen. The ai detectives, 2017.

Sandra Wachter, Brent Mittelstadt, and Chris Russell. Counterfactual explanations without opening the black box: Automated decisions and the gdpr. *arXiv preprint arXiv:1711.00399*, 2017.

Wenjia Wang, Phillis Jones, and Derek Partridge. A comparative study of feature-salience ranking techniques. *Neural Computation*, 13(7):1603–1623, 2001.

Leanne S Whitmore, Ryan W Davis, Robert L McCormick, John M Gladden, Blake A Simmons, Anthe George, and Corey M Hudson. Biocompoundml: A general biofuel property screening tool for biological molecules using random forest classifiers. *Energy & Fuels*, 30(10):8410–8418, 2016.

Leanne S Whitmore, Anthe George, and Corey M Hudson. Mapping chemical performance on molecular structures using locally interpretable explanations. *arXiv preprint arXiv:1611.07443*, 2016.

Qifeng Zhou, Wencai Hong, Linkai Luo, and Fan Yang. Gene selection using random forest and proximity differences criterion on dna microarray data. *Journal of Convergence Information Technology*, 5(6):161–170, 2010.