Applying the CASSM Framework to Improving End User Debugging of Interactive Machine Learning

Marco Gillies
Embodied Audio-Visual Interaction, Department of Computing, Goldsmiths' University of London
London SE14 6NW
m.gillies@gold.ac.uk

Andrea Kleinsmith
Department of Computer and Information Science and Engineering
University of Florida,
Gainsville, Florida, USA
alk@cise.ufl.edu

Harry Brenton
Embodied Audio-Visual Interaction, Department of Computing, Goldsmiths' University of London
London SE14 6NW
h.brenton@gold.ac.uk

ABSTRACT

This paper presents an application of the CASSM (Concept-based Analysis of Surface and Structural Misfits) framework to interactive machine learning for a bodily interaction domain. We developed software to enable end users to design full body interaction games involving interaction with a virtual character. The software used a machine learning algorithm to classify postures as based on examples provided by users. A longitudinal study showed that training the algorithm was straightforward, but that debugging errors was very challenging. A CASSM analysis showed that there were fundamental mismatches between the users concepts and the working of the learning system. This resulted in a new design in which aimed to better align both the learning algorithm and user interface with users' concepts. This work provides and example of how HCI methods can be applied to machine learning in order to improve its usability and provide new insights into its use.

Author Keywords

Interactive Machine Learning; Movement Interfaces; Gesture Design; Conceptual Models;

ACM Classification Keywords

H.5.2. Information interfaces and presentation (e.g., HCI): User Interfaces: Evaluation/methodology

General Terms

Human Factors; Design; Experimentation.

INTRODUCTION AND RELATED WORK

Full body and gestural interfaces can be a compelling and engaging way of interacting with games and other software[1]. One important application of full body interfaces is the ability to interact with an animated virtual character using natural body language, examples range from Thórisson's early work [7] to Huang et al's [5] sophisticated responsive listening agent. Recent research has focused on how to support end users in designing their own bodily and gestural interfaces[4, 8]. These methods use Interactive Machine Learning (IML) [3] in which users provide examples of interactions that are used to train a machine learning classifier. This has the benefit that users can create interaction systems by performing movement without having to analyze it into low level components of features [4]. However, this does not mean that this approach is automatically easy to use. Machine learning algorithms can be very complex and it can be difficult for end users to build adequate conceptual models of how they work.

In this paper we present a system that users were able to use effectively to train a system, however, debugging the system when it did not work effectively was challenging. This is interesting as debugging relies greatly on effective conceptual models (an issue also highlighted by Kuleaza et al.[6]). We applied CASSM (Concept-based Analysis of Surface and Structural Misfits) framework by Blandford et al [2] to understanding our users' difficulties and redesigning the system to better support them. CASSM is a method of evaluating the concepts inherent in a piece of software and how they may mismatch with users' concepts, and so better support users in creating effective conceptual models. The method involves identifying concepts that are important to users and present in a system and identifying how these concepts match between users' understanding, the software interface and the system itself. In each of these three elements a concept may be "present", "absent" or "difficult", meaning that the concept is not straightforward to understand or mismatched relative to another element.

USER CENTRED DESIGN

We performed a longitudinal user centred design process to inform the design of our system. The process was driven by one expert user: an actor and physical performer. The aim of this process was to understand how a body movement expert, with no computer science training, might learn to use Interactive Machine Learning software. We designed and tested a software platform to allow him to design a game in which a player interacts with a virtual character via body movements. The performers task was to design a character from scratch. This character would interact with with a game player via body movement. During the user centred design process the actor worked with 8 non-expert participants (all students in

IUI'15, March 29 - April 01, 2015, Atlanta, GA, USA

ACM 978-1-4503-3306-1/15/03 ...$15.00.
http://dx.doi.org/10.1145/2678025.2701373

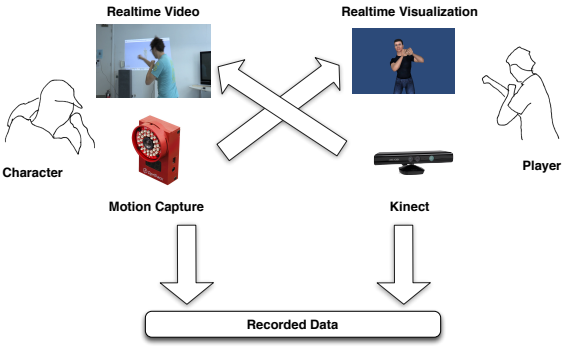

Figure 1. The capture process.

the Department of Computing at Goldsmiths, University of London, 6 male and 2 female) who played the part of the player, either interacting directly with the performer or with our software.

In the initial phase of the study we observed the performer interacting with two of the participants (both male) to design the character's behavior. The core of the performer's method was identifying actions by the player and responses by the character, for example, the player looks threatening and the character cowers in fear. Action-response pair is a central concept in how he organized his work. In preparation for a session he would write down a sequence of pairs (e.g. "applaud-bow", "punch-fall") and the study included a lot of discussion of the suitability and effectiveness of individual pairs. This approach seemed to map naturally onto a machine learning methodology, in which a classifier could be used to classify actions from the player and the results used to generate responses in the character. The examples of actions and responses performed by the participants during improvisation could serve as training data for the classifier.

These results were used to design software to support the design of the character. It used the concept of action and response as the basis of a classifier based machine learning system described above. In the rest of the paper we will use the term **action** to describe a meaningful movement performed by the player and **response** for the movement to be performed by the virtual character in response to the action. Responses are represented as short clips of animation data extracted from the motion capture. Actions are detected using a machine learning classifier. The purpose of the classifier is to analyse novel movement data from the player and determine which of a number of discrete classes it belongs to. Each class is designed to correspond to a type of action. However, the two will not be in perfect correspondance the classifier will sometimes classify movements in ways that do not correspond with the performer's understanding of a particular type of action. This distinction is critical as we shall see later in the paper. In the rest of the paper we will use the term **class** to mean what it learned by the machine learning classifier and **type of action** to mean the user's concept.

Figure 1 shows outlines the process for designing a virtual character. Two people, one acting the part of the character (the performer), the other acting the part of the player, interact with each other video a live link. The performer saw a live video of the player while the player saw the performer represented as a 3D virtual character. Their movement is recorded, the character via optical motion capture, the player via a Microsoft Kinect^{TM}. The resulting data is then accessible via a labeling interface (figure 2, left). Users can select sequences of the character's movements together with the corresponding action by the player. These are used to train a Decision Tree classifier which learns a mapping between player's actions and the character's responses. The performer used this interface over 3 sessions each lasting about 3 hours. He designed a character and tested it with 6 of the participants (2 female, 4 male) acting the part of the player.

RESULTS

There were a number of positive outcomes of these studies. Firstly, the performer and other participants were able to follow the process and found the interface quick to learn, as the performer commented: *"technically it is quite easy to learn, you learn it fast. All the people that came in after a few seconds everybody was able to add clips, add classes"*. In all sessions, the performer created an interactive character and was able to test it. Some responses were very successful. For example, one participant commented *"It was very good at ... when I laughed at it, it would walk away disappointed"*.

However, overall, the performance of the interactive character was disappointing. While some actions were recognized reliably others were not. As one participant describes *"about 60% of the time it did what I wanted it to do. The other 40% of the time it was insulted"*. What is more the performer and the other participants were not able to clearly identify specific causes of problems (*"about the clips I'm still not clear what ... the problem [is]"*) and therefore any proposed solutions were vague and often inappropriate. The performer proposed solutions mostly related to the physical performance rather than the labeling process. Examples of his ideas include being aware of the body shapes of participants (he suggested that one session worked better because *"the two people we worked with today had a more similar physicality"*) or taking more control himself by playing both parts (*"I would like to be the block man too"*). This shows that while he was well aware that the performance was poor but was not able to improve it. This shows that the most difficult task was debugging the model when it did not work correctly, a key issue for IML as noted by Kulesza et. al.[6]. Debugging requires users to have a correct conceptual model of the system, and how it might fail, in order to formulate an appropriate solution. Making a system easier to debug therefore means supporting users in creating better conceptual models of software.

We applied the CASSM methodology to analyzing the conceptual model implicit in the software. Table 1 shows a simplified CASSM analysis of the first prototype. It only covers concepts and not the action elements of the CASSM framework nor does it distinguish the types of concept (entity, attribute) proposed in CASSM. While the generic CASSM framework includes three loci at which concepts can be represented (user, interface and system) we focus on analyzing

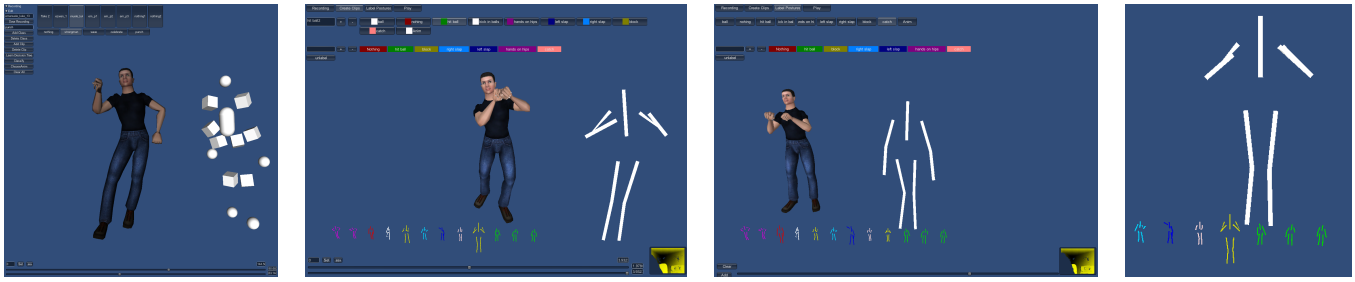

Figure 2. The evolution of the user interfaces. From left to right: the labeling interface of the first prototype the layout of the second prototype of our software which more clearly separates out different aspects of the process. Left, the screen for selecting clips; the screen for labeling feature data; the dynamic visualization of the nearest neighbor algorithm. Each data item is displayed as a stick figure. The figures are scaled in proportion to the amount of probability each contributes to the classification of the current posture.

concept	user	interface	classifier
action	present	*difficult*	*absent*
response	present	present	present
a-r pair	*difficult*	present	*absent*
class	present	present	present
feature	*difficult*	*absent*	present

Table 1. A CASSM analysis of the first prototype

the classifier rather than the system as a whole as this is the element of the system where the difficulty seems to lie and analyzing it separately allows us to identify conceptions such as "action" that are present in the system as a whole but (problematically) absent in the classifier.

The key finding of the first phase of this study was that the performer's key working concepts were actions and responses. Responses are clearly represented in the interface as animation clips that are selected by users and also in the classifier as sets of clips that can be played in response to a particular classification. However, there are a number of conflicts between participants concepts of actions and how they are represented in the system.

While actions and types of action are central to users' understanding of the process and the interface, they are not well represented in the classifier itself. The classifier used was a decision tree which uses the data from individual actions to train a generic model, and once no longer uses them once it has done so. The classifier does not even use the concept of a holistic pose in its classification, but makes a sequence of decisions based on individual features (e.g. left shoulder rotation about the x-axis). The central concept of the classifier is therefore the feature, not the whole action. This conflicts with the interface, which deals in action and also with the performer's working practices. While users were able to think about individual features to a degree (or at least individual body parts), most of the discussions between the performer and other participants were at the level of holistic actions; discussions of individual body parts were rare.

A second mismatch relates to pairing actions and responses. As noted above, action-response pairs were central to the performer's way of work in the project, for this reason the action-response pair concept was built in to the interface. Users provide training data by selecting an action-response pair from the recorded data. A section of data represents both an action (the Kinect data at the start of the section) and a response (a clip formed of the entire selected motion capture data). However, in the classifier action data and response data are treated entirely separately. Presenting action-response pairs therefore conflates two concepts which are distinct within the system and discourages users from considering them independently. What happened initially was that the performer and other participants concentrated primarily on responses rather than actions. The performer said that *"I didn't know about the block man, I understood the block man towards the real end"*. Pairing actions and responses can make it more challenging to train an effective model as it may not be possible to find clips that are good both as training data and as animation clips. Action-response pairs are not the only concepts that link actions and responses. The performer did not think solely in terms of individual instance of actions and responses but in types of action and response (such as "applaud" and "punch"). A generic type of response was linked to a generic type of action, but the individual response movement were not specific to an individual action movement. Action type - response type pairs are therefore the key concept, not action-response pairs. Types of action are well represented in both the interfaces and classifier in the form of classes. The pairing of types of action with classes in the classifier therefore forms the major bridge between the perfumer's working practice and the machine learning algorithm.

REDESIGNING THE PROTOTYPE

The CASSM analysis was used to inform a redesign of the system, with a new interface shown in figure 2 (center). As noted, the classification algorithm used is key in supporting or not supporting user concepts. Changing the algorithm can have an important effect on how understandable the system is. Classification algorithms embody concepts of the domain that can either match or mismatch with user concepts. Mismatched concepts can lead to situations in which the algorithm does not effectively support users or in which users are not able to use the algorithm effectively. Concepts that match closely are more likely to support users' work, but even systems that are close to user concepts will not immediately be understood unless they are communicated effectively in the interface. Therefor all of the concepts we have discussed need to be well represented both in the classifier and the interface.

The new classifier uses a weighted nearest neighbor method. Nearest neighbour methods directly use the examples in the original dataset, new items are classified based on the most similar training examples. A standard nearest neighbor algorithm selects a fixed number of neighbors (e.g. 3) and treats them equally. The classification will be the majority vote of the neighbors. On the other hand a weighted approach applies a weight to the influence of a data example based on it's distance form the item to be classified. Our classifier uses all the examples but weights them by distance. A weighted approach is likely to work better when there are a small number of data items. If the data is very sparse, a standard nearest neighbor approach is likely to select neighbors that are very different from the item to be classified. If these are all treated equally, they are likely to result in poor performance. If they are weighted by distance, they are less likely to affect the performances. The use of a weighted algorithm was also motivated by the intuition that users are more likely to think in terms of examples being more or less similar rather than in terms of an arbitrary cut off where some examples are similar and some are not.

The classifier was implemented as a Gaussian Mixture Model, a probabilistic model that gives an estimated probability of a data item being in each class. The probability is the sum of a number of multi-variate Gaussian distributions:

$$P = \sum_i \mathcal{N}(\mu_i, \sigma_i)$$

μ_i is the mean of the i^{th} distribution and σ_i is it's variance. Each class has it's own Gaussian Mixture Model which gives the probability of an item being in that class. There is a Gaussian for each training example that belongs to that class. The mean of that distribution is the feature vector for that example. The Gaussian therefore gives an exponentially weighted measure of the distance from each example and therefore particularly privileges similar examples, and virtually ignores distant ones. σ_i is the same for all examples and is spherical (i.e. it is the same for all dimensions of the feature vector). It could be used as a tunable parameter, or varied between examples, to give greater weight to certain example, but we did not explore these possibilities in this prototype.

This new algorithm aims to align the working of the algorithm with users' conceptual models, because actions are represented directly. It is likely to perform better with a small number of data items, as long as they are well chosen. More importantly, it makes it possible to provide better visual feedback to users about the functioning of the algorithm. As the classifier is defined by all of the training examples, it can be visualized simply by presenting all of the actions used to train the system. This visualization should map very directly onto user concepts as it directly displays the actions they have selected. Figure 2, far right, shows the new visualization. Each training example is represented using the same stick figure representation as is used for the live and recorded Kinect data. The examples are color coded to represent classes (the same colors are used in the class buttons shown in Figure 2, center). In order to give real time feedback about the system, the visualization needs to represent the reasons why the classifier is classifing particular instances the way it does. To do this visualization shown in Figure 2, far left, scales each example in proportion to the weight assigned to that example by the classifier (plus a constant to ensure that even components that contribute zero weight are still visible). This scaling is done during live testing, but also on the labeling screen, so users can see how the classification changes as they scrub through their data.

Other mismatches can be solved by changing the interface rather than the classifier. The first interface was built around the concept of action-response pair that is not present in the classifier. The second prototype separates the two concepts with different interfaces for selection response clips and labeling feature data containing actions. (figure 2 center left and center right). This separation aimed to make the distinction conceptually clearer. It also allowed users to select the most suitable clips and Kinect poses independently, and therefore be able to select the best of each.

CONCLUSIONS

This paper has presented an example of how a CASSM analysis can support researchers in understanding difficulties in an Interactive Machine Learning interface and redesigning that interface. It was particularly useful in understanding how machine learning can fit into users conceptual understanding. CASSM is valuable here because it forces designers to think about conceptual models not simply in terms of the interface but also at the level of machine learning system, which is all too often treated as a black box. While a black box approach can seemingly simplify the user experience, it can lead to confused conceptual models and hinder users when they must debug the learning system. CASSM's focus on how conceptual models map into the computational system may be able to help overcome this problem by encouraging designers to align the learning algorithm used with users' conceptual models. However, it is important to note that analysis like CASSM, while it can result in a plausible design, is not sufficient to demonstrate that this design is actually usable in pracitce. Considerable further work is needed. The next step is to perform a large scale controlled trial to test whether the system is usable. This work will no doubt result in many changes and iterations before a fully usable interactive machine learning system is achieved.

More generally, this work is an example of how established HCI methodologies can be applied to Machine Learning, to improve its usability but also to provide new ways of understanding learning algorithms: not simply in terms of efficiency or accuracy but also how they are understood by users. This is an important step in making Machine Learning accessible to a wide public, beyond the academic and industry research labs to which it is currently largely confined.

ACKNOWLEDGMENTS

This work is supported by the UK Engineering and Physical Sciences Research Council, under grant EP/H02977X/1, Performance based expressive virtual characters.

REFERENCES

1. Bianchi-Berthouze, N., Kim, W. W., and Patel, D. Does body movement engage you more in digital game play? and why? In *Proceedings of Affective Computing and Intelligent Interaction, ACII 2007*, A. Paiva, R. Prada, and R. W. Picard, Eds., vol. 4738 of *Lecture Notes in Computer Science*, Springer (2007), 102–113.
2. Blandford, A., Green, T. R. G., Furniss, D., and Makri, S. Evaluating system utility and conceptual fit using cassm. *Int. J. Hum.-Comput. Stud. 66*, 6 (June 2008), 393–409.
3. Fails, J. A., and Olsen, Jr., D. R. Interactive machine learning. In *Proceedings of the 8th international conference on Intelligent user interfaces*, IUI '03, ACM (New York, NY, USA, 2003), 39–45.
4. Fiebrink, R., Cook, P. R., and Trueman, D. Human model evaluation in interactive supervised learning. In *Proceedings of the 2011 annual conference on Human factors in computing systems*, CHI '11, ACM (New York, NY, USA, 2011), 147–156.
5. Huang, L., Morency, L.-P., and Gratch, J. Learning backchannel prediction model from parasocial consensus sampling: a subjective evaluation. In *Proceedings of the 10th international conference on Intelligent virtual agents*, IVA'10, Springer-Verlag (Berlin, Heidelberg, 2010), 159–172.
6. Kulesza, T., Stumpf, S., Wong, W.-K., Burnett, M. M., Perona, S., Ko, A., and Oberst, I. Why-oriented end-user debugging of naive bayes text classification. *ACM Trans. Interact. Intell. Syst. 1*, 1 (Oct. 2011), 2:1–2:31.
7. Thórisson, K. Real-time decision making in multimodal face-to-face communication. In *second ACM international conference on autonomous agents* (1998), 16–23.
8. Zamborlin, B., Bevilacqua, F., Gillies, M., and D'inverno, M. Fluid gesture interaction design: Applications of continuous recognition for the design of modern gestural interfaces. *ACM Trans. Interact. Intell. Syst. 3*, 4 (Jan. 2014), 22:1–22:30.