Confidence, command, complexity: metamodels for structured interaction with machine intelligence

Advait Sarkar

Computer Laboratory, University of Cambridge
Wiliam Gates Building, 15 JJ Thomson Avenue, Cambridge, UK
advait.sarkar@cl.cam.ac.uk

Abstract. Programming is a form of dialogue with machines. In recent years, we have become increasingly involved in a dialogue that shapes our surroundings, as we come to inhabit a newly inferred world. It is unclear how this dialogue should be structured, especially as the notion of "correctness" for these programs is now unknown or ill-defined. I present a speculative discussion of a potential solution: metamodels of machine cognition.

Keywords: POP-II.B. Program Comprehension; POP-IV.B. User Interfaces; POP-I.C. Ill-Defined Problems

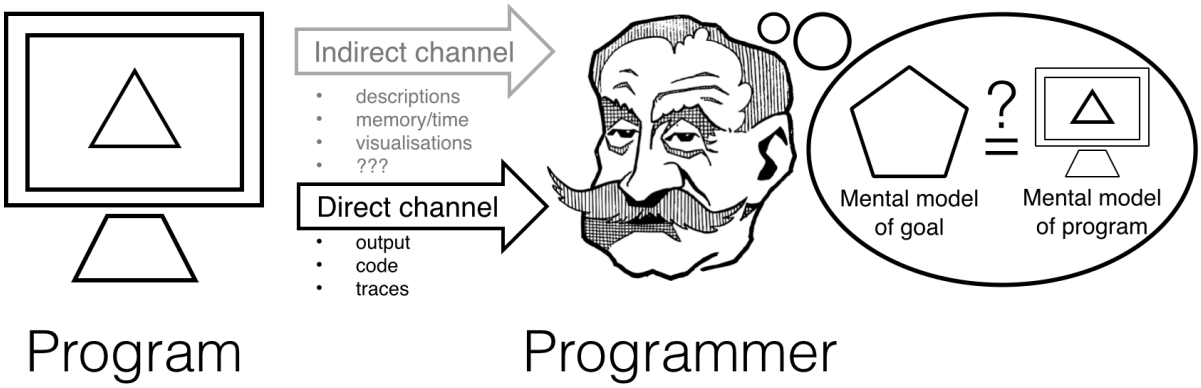

Fig. 1. *Are you thinking what I'm thinking?* The old paradigm; programmer and program communicate primarily through direct channels of inspection.

1 Introduction: a paradigm shift in programming

We increasingly inhabit an inferred world. The dominant mode of programming is changing. To explain more clearly, I shall first paint a simplified caricature of the traditional programming paradigm. Figure 1 shows a diagram, representing the traditional interaction between programmer and program. Here, the programmer has a goal mental model of the information structure to be built. Through a direct channel, such as inspection of the source code, its output, and execution traces, the programmer is able to build a mental model of the information structure as it currently is. Thus, the programmer is able to compare these two models against each other in order to decide whether the program matches the goal, whether it is incomplete, or whether it contains errors.

The old paradigm is characterised by the utility of the explicit data in the direct channel. The expected output is sufficiently well-defined that should the output depart from the programmer's expectations (i.e., an error), an inspection of the workings of the program will suffice to resolve the situation (i.e., debugging). A great deal of study (§2) has been conducted on enriching the debugging experience with implicit information through the "indirect" channel, for example, through descriptions of the program, through inspections of its time and memory requirements, and through visualisations of its operation. Nonetheless, it is still possible, and in many cases sufficient, to conduct debugging through direct inspection of the program source code, output, and traces. Thus, the activities surrounding the traditional programming paradigm can be summarised as "are you thinking what I'm thinking?"

1.1 End-user machine learning is the new programming

Programs are different now, however. We increasingly inhabit an inferred world (Blackwell, 2015), and the outcome of computer algorithms is becoming predominantly probabilistic and data-dependent, rather than deterministic.

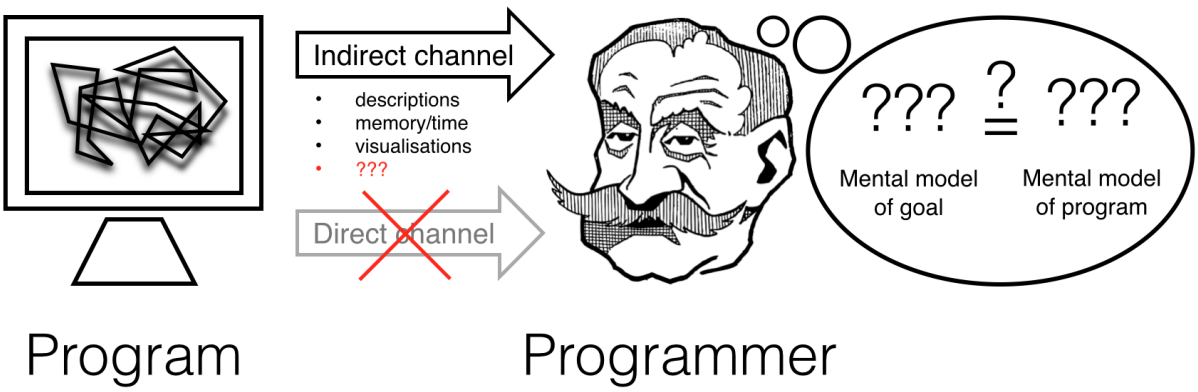

Fig. 2. *What are we thinking?* The new paradigm; in the absence of a useful direct channel, we must structure our dialogue around the indirect.

The training of machine learning models can be regarded as as an act of programming. End-users of systems such as recommendation systems (e.g., Amazon's product recommendations, Pandora's music recommendations), intelligent personal assistants (e.g., Apple's Siri, Microsoft's Cortana, Google Now), and intelligent consumer tools (e.g., Excel's Flash Fill) etc. increasingly find themselves implicitly or explicitly programming their environment. However, our interaction with these systems largely remains opaque to their decision making process, which often involves considerable uncertainty. When the output departs from our expectations, neither are our expectations well-defined, nor does inspecting the workings of the system resolve the situation – this is where a dialogue is necessary.

Figure 2 illustrates how the inferred world has shifted the predominant programming paradigm. Programming in the new paradigm is characterised by the following three properties:

1. The "programs" are stored as massive quantities of model parameters, and thus are largely human-unintelligible.
2. The programmer is likely to be an end-user programmer who is not necessarily skilled at computing.
3. The goal state of the program is unknown or ill-defined.

The combination of these makes the traditional direct channel inapplicable, and can be summarised as a "what are we thinking?" approach to programming, where mental models of neither goal nor program are well-structured. As a consequence, we must place greater care with the way we exploit the indirect channel, that is, we must shift from an emphasis on facilitating the user's understanding *of the* program to their understanding *about the* program. Previous debugging literature has by no means ignored this channel, and neither has the interactive machine learning literature. However, treatment of this channel has typically been on an ad-hoc basis. By explicitly acknowledging the interaction as dialogue, we are able to take a structured approach, which is descriptive as well as prescriptive.

In this paper, I propose a fundamental addition to the indirect programming channel: *metamodels of machine cognition*. Machine learning consists of algorithms which model their output as functions of their input. But the output of a machine learning model alone does not suffice for a rich interactive dialogue. Is the model confident in its own output? Has the model had adequate exposure to the domain? If these were known, we might be able to critically appraise its predictions in a wider context. We might be able to direct the learning of the model, to expose it to parts of the domain it still does not know about, or to provide appropriate training data to help improve its confidence. How complex was the prediction to make? If this was known, we might be able to spot and rectify trivial simplifications of the target domain that the algorithm is exploiting in order to make predictions.

2 Related areas of inquiry

2.1 Interactive machine learning

Our primary application domain of interest is the field of interactive machine learning. An early exploration of how a visual interface might enable end-users to effectively build classifiers is given by Ware et al. (2001), describing the graphical interface for the popular Weka machine learning toolkit. However, their application was still very much directed towards expert statisticians. Subsequently, the work in the area has become focused on end-users with less awareness of statistical and computing concepts.

Perhaps the archetype of the field is the eponymous paper demonstrating the *Crayons* application (Fails & Olsen, 2003), where users could train a classifier for image segmentation by directly sketching over parts of the image to indicate positive or negative examples. Fogarty et al. (2008), and more recently Kulesza et al.

(2014) tackle the problem of labelling concepts in images, where "concepts" are not always predefined classes, but rather can be evolved over the course of the labelling exercise. Fiebrink et al. (2011) demonstrate interactive model training for realtime music composition and performance. To some extent, one can also consider the following to be examples of interactive machine learning: Brown et al. (2012), who demonstrate a visual interface for specifying distance functions, and Hao et al. (2007), who show how data visualisations can be used as a querying interface.

Fails and Olsen motivate their work by emphasising the ease of generating a classifier in an interactive visual manner. Similarly, Fogarty's, Brown's and Hao's systems are presented from primarily an ease-of-use view. My own work in end-user machine learning in spreadsheets (Sarkar et al., 2014) focuses on ease-of-use. These systems achieve ease of use by massively abstracting away the workings of the system, which is generally a useful strategy, as long as the behaviour of the program corresponds to user expectations. But what happens when the system gets it wrong, and not in a way that is easily apparent (Szegedy et al., 2013; Nguyen et al., 2015)? To better involve the user in the process, the repeated use of the word "explain" throughout the interactive machine learning literature (Herlocker et al., 2000; Tintarev & Masthoff, 2007; McSherry, 2005; Pu & Chen, 2006) does not appear to be coincidental; clearly the underlying aim is to give our interactions with programs much more of a dialogue-like quality.

A critical assessment of end-user interaction with machine learning has been made by Amershi et al. (2011). The authors identify a few questions for this dialogue: *What examples should a person provide to effectively train the system?*, *How should the system illustrate its current understanding?*, and *How can a person evaluate the quality of the system's current understanding in order to better guide it towards the desired behavior?* Systematic metacognitive modelling provides a partial answer to all of these questions.

Lim & Dey (2009) have directly addressed the problem of what types of information about intelligent applications should be given to end-users. They call these "intelligibility types," and some examples of these are as follows: *Input & output*: what information does the system use to make its decision, and what types of decision can the system produce? *Why, why not, & how*: why did the system produce the output that it did, why did it not produce a different output, and how did it do so? *What if*: what would the system produce under given inputs? *Model*: how does the system work in general? *Certainty*: how certain is the system of this report? *Control*: how can I modify various aspects of this decision making process? Kulesza et al. (2013) show that these information types are critical for the formation of users' mental models. Kulesza et al. (2011) also proposed a set of information types which would benefit end-users who were debugging a machine-learned program, including: *Debugging strategy*: which of many potential ways of improving the model should be picked? *Model capabilities*: what types of reasoning can the model do? *User interface features*: what is the specific function of a certain interface element? *Model's current logic*: why did the model make certain decisions? *User action history*: how did the user's actions cause the model to improve/worsen? This presents excellent motivation for systematic metacognitive modelling, without which such information cannot be generated.

2.2 End-user debugging

The producers of these machine learning models are also their users. As such, this is related to end-user software engineering (Ko et al., 2011), and in particular end-user debugging. Interestingly, end-user debugging has so far been quite explicit in framing the interaction as dialogue. Wilson et al. (2003) argue that programming assertions in spreadsheets is difficult and boring, and present a strategy to incentivise users to write more assertions. This strategy – surprise, explain, reward – is much like dialogue. The software generates what it thinks is a surprising assertion that nonetheless fits a cell's formula. It changes the value of the cell to be valid under this assertion, and explains this decision and how to change the assertion through a tooltip. Finally, the user is "rewarded" by virtue of having a more correct spreadsheet.

Ko & Myers (2004) present the "WhyLine," a debugging tool that is meant to operate literally as dialogue. By scanning the function call structure of a program execution, the tool can create hierarchical menus which allow the user to formulate grammatically correct "why" questions about the execution of a program. Interestingly, Kulesza et al. (2009, 2011) take this approach to facilitate end-user debugging of the underlying naïve Bayes model of a email spam classifier.

As with interactive machine learning, allusions to "explanations" also appear throughout the end-user debugging literature. However, there is an important distinction to be made between the type of dialogue one engages in when debugging, and the type of dialogue one has with a machine learning model. The activity of "debugging" principally occupies the direct interaction channel, as in the old paradigm. Treating the training of machine learning models as debugging can only be informative for interaction design up to a point. The debugging situation assumes that the user's mental information structure is the correct version, which the computer's internal information structure must aim to reproduce. That is, we assume the human knows the right answer. This is not to say that the programmer always knows how to concretely express the required information structure in a given programming language; perhaps the programmer receives assistance from the system, as in WYSIWYT (Rothermel et al., 1998). However, in the old paradigm, the final arbiter of what is, and is not a "bug," is the programmer.

Conversely, in the new paradigm, the right answer is either unknown or ill-defined. It follows that under these circumstances, "debugging," or even a "bug," cannot definitively exist. In the class of situations we are dealing with today: product recommendations, automated diagnostics, weather forecasting, etc., neither the human nor the computer knows the right answer, but rather they are in a dialogue to try and resolve the issue together. Thus, both parties must be transparent to one another. I suspect that one of the reasons Teach and Try (Sarkar et al., 2014) was so successful at generating an understanding of statistical procedures in non-experts is the deliberate selection of the word "Try" as opposed to "Fill" or "Apply model"; it implies fallibility and evokes empathy.

2.3 Mixed-initiative interaction

Mixed-initiative interaction explicitly acknowledges that program behaviour could be usefully augmented by models that were not strictly about the problem domain. In this case, the models being made are of the user, and of user intent. Horvitz (1999) argues that mixed-initiative systems (i.e. automated services) must exhibit certain "critical factors", or principles. The most pertinent of these to this paper is that decisions must be made under awareness of uncertainty about user goals, and the cost of distracting the user.

As a case study, Horvitz uses a calendaring service which automatically parses emails for event date/time information and suggests actions based thereupon. Importantly, Horvitz provides a decision-theoretic heuristic for taking an action based on an expected utility function. This function is calculated given beliefs about a user's goals derived from observed evidence. Action is taken when the utility for action exceeds that of inaction. In order to implement this, an explicit utility model must be built and updated as the user interacts with the software. This idea can be adopted for our use in the new programming paradigm, not to model the user, but to model the program itself.

3 A proposition: models of machine cognition

From interactive machine learning, I take the pertinent and emergent domain of end-user programmers of machine learning models. From end-user debugging, I adopt the strategy of treating interaction as dialogue. From mixed-initiative interaction, I appropriate the strategy of developing explicit metamodels, to consider thinking *about the* program rather than *of the* program itself. Consequently, I propose that it is a useful, systematic strategy to augment machine learning models with metamodels. What should be the subject of these metamodels, and how many are required? Let us begin with the following:

1. **Confidence**: how sure is the program that a given output is correct?
2. **Command**: how well does the program know the domain?
3. **Complexity**: did the program do a simple or complex thing to arrive at the output?

I believe that these three are necessary for successful dialogue of the kind outlined in the introduction. They are not exhaustive, but have emerged to be clearly important from careful consideration of the engineering requirements for improving end-user programming of machine learning models in a variety of scenarios, which shall be elaborated in §4.

The metamodels are intimately related to the information types proposed by Lim & Dey (2009) and Kulesza et al. (2011). Those frameworks prescribe types of information which would be beneficial to an end-user programmer of machine learning models, but do not prescribe *how* such information might be generated. So while these metamodels are a conceptual solution at the same level as the intelligibility types, i.e., they prescribe things which should be shown to the user, they are also an engineering solution at a technical level, i.e., they prescribe how this information can be generated. With systematic metamodelling, it may not be necessary to recreate methods for providing intelligibility for each new interface and machine learning system on an ad-hoc basis.

In the following subsections, I shall illustrate and elaborate upon each of these three metamodels in turn.

3.1 Confidence

Confidence has been dealt with throughout the statistics and machine learning literature. Methods for estimating the error or confidence for any given output have been developed for many models. Linear regression, one of the simplest statistical models, is accompanied by a procedure for computing the 95% confidence intervals for its learnt parameters, which can be interpreted as confidence: the narrower the intervals, the more confident the prediction. However, being able to estimate this confidence is not necessarily incentivised in benchmarks of machine learning performance, which are primarily concerned with the correctness of the output.

Table 1 presents some suggestions for how confidence may be computed for popular machine learning techniques. Measures of confidence can be used to prioritise human supervision of machine output; when there are large quantities of output to evaluate, the user's attention can be focused on low-confidence outputs, which may be problematic. González-Rubio et al. (2010) use this approach to improve interactive machine translation, and

Kulesza et al. (2015) use this approach to improve interactive email classification. Behrisch et al. (2014) show a vision of enriched dialogue, made possible through a confidence metamodel: in their software, the user interactively builds a decision tree by annotating examples as "relevant" or "irrelevant," but is able to decide when the exploration has reached convergence due to a live visualisation of how much of the data passes a certain threshold for classification confidence.

Table 1. Practical confidence metamodel suggestions

Model	**Suggested calculations of confidence**
k-NN	For a given prediction, confidence can be measured as the mean distance of the output label from its k nearest neighbours as a fraction of the mean pairwise distance between all pairs of training examples. A similar metric is proposed in Smith et al. (1994).
Neural Network	For a multi-class classification, where each output note emits the probability of the input belonging to a certain class, confidence can be measured simply as the probability reported. More sophisticated confidence interval calculations can be obtained by considering the domain being modelled, as in Chryssolouris et al. (1996); Weintraub et al. (1997); Zhang & Luh (2005)
Decision Tree	The confidence of a decision tree in a given output can be measured as the cumulative information gain from the root to the outputted leaf node. Alternatively, Kalkanis (1993) provides a more traditional approach.
Naïve Bayes	The confidence of a Naïve Bayes classifier in a given prediction can be measured as the probability of the maximally probable class. More sophisticated treatment of the problem is given by Laird & Louis (1987); Carlin & Gelfand (1990).
Hidden Markov Model	The primary tasks associated with HMMs (filtering, prediction, smoothing, and sequence fitting) all involve maximising a probability; the confidence can simply be measured as the probability of the maximally probable output. More fine-grained confidences can be measured by marginalising over the relevant variables (Eddy, 2004).

Confidence alone, however, can be deceiving. Recent work (Szegedy et al., 2013; Nguyen et al., 2015) has demonstrated how some apparently straightforward images with carefully injected noise, as well as completely unrecognisable images, are still classified with high confidence by a state-of-the-art image classifier. Thus, confidence is not the end of the story when it comes to understanding a machine's abilities – it may be necessary, but is not sufficient.

3.2 Command

Addition of a second metamodel, "command," is a further step towards enriching the description of machine understanding. It has been expressed in various forms in the literature. The dream of a self-regulated, autonomous agent is long lived in GOFAI and modern machine learning, motivated by such issues as the "exploration-versus-exploitation" tradeoff; i.e., should the agent do something which has been known to provide a certain reward, or should the agent explore the wider world in search of potentially better rewards, at the risk of wasting resources on less-rewarding world states?

Systems developed towards this aim often exhibit primitive forms of metacognition. A most basic example of a famous problem which benefits from this form of metacognition is that of the multi-armed bandit (Gittins et al., 2011). A gambler at a row of slot machines has to decide which machines to play, how many times to play each machine, and in which order to play them, in order to maximise the cumulative reward earned. Each machine provides a random reward from a distribution specific to that machine. Thus, the tradeoff is between *exploration*, i.e., playing machines in order to learn about their reward distributions, and *exploitation*, i.e., playing machines in order to gain the reward. A solution to this problem must necessarily involve a model of command, i.e., how much is known about the reward distribution of each machine, in order to effectively navigate this tradeoff.

Similarly, the concept of reinforcement learning (Watkins, 1989) involves a "reward function", which records the reward an intelligent agent might hope to receive upon transitioning to any given world state; the agent can then probabilistically transition to world states that will either fulfil its information need by updating the reward function, or alternatively will pay off by way of actually receiving the reward. A related concept is *active learning* (Cohn et al., 1996; Settles, 2010), where the algorithm is able to select examples it believes to be most useful for its learning, and presents these examples to a human oracle (or other information source) for labelling. The motivation behind active learning is similar to exploration-vs-exploitation: that the algorithm may achieve greater accuracy with fewer training examples should it choose the data from which it learns. Savitha et al. (2012) show a "metacognitive" neural network which can decide for itself what, when, and how to learn from each training datum it is given. Interestingly, common to these techniques is their reliance on an additional model, that of the input domain, so that the agent is able to distinguish between what is known and what remains to be known. In

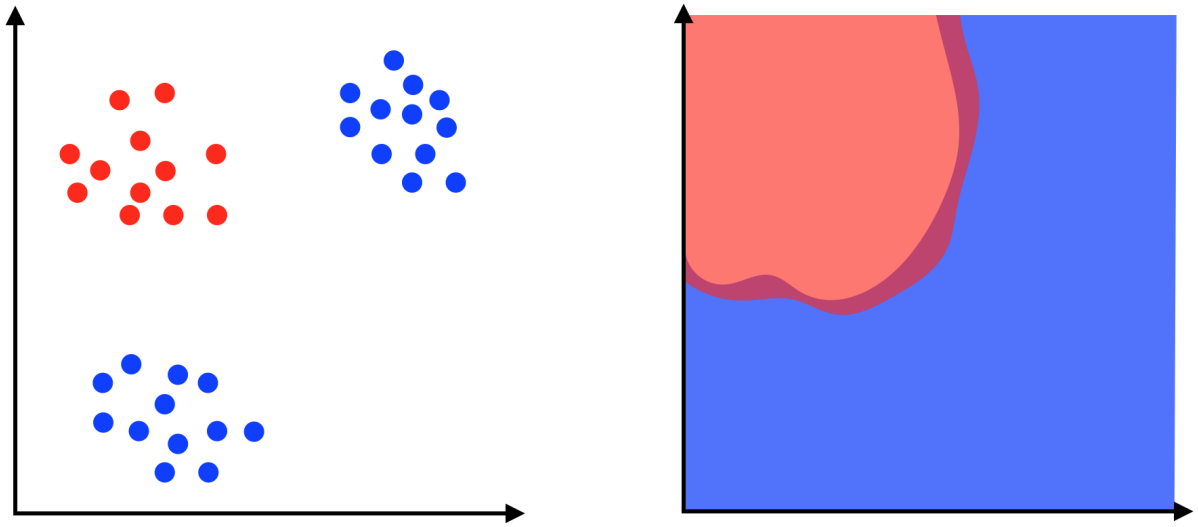

Fig. 3. Two alternate views of command: on the left, a visualisation of labelled training examples in the input space. On the right, a visualisation of the learned decision boundaries, showing an area of reduced certainty. These are radically different interpretations. Consider the bottom right-hand corner. The classifier predicts 'blue' with high confidence, but since it has not seen any examples from that area, should it really be confident?

reinforcement learning, this takes the form of the *state space*. Thus, any practical definition of "command" has to be constructed in relation to a definition of the domain being modelled.

I can suggest two simple methods of illustrating the command of a machine learning algorithm over a certain domain. The first is to look at the training examples the classifier has so far received, as positioned in the input domain. The second is to look at the classifier's confidence at all points in the domain. These are illustrated in Figure 3. It is clear that these two images paint a very different picture of the algorithm's "command" over a domain. If we view command as some integral of confidence, then an algorithm with high levels of confidence in the majority of the domain can be considered to have a good command of the domain. If we view command as some integral of the occurrences of training examples encountered, then an algorithm which has received a uniform spread of training examples may be considered to have a good command of the domain.

The command metamodel is intimately related to the problem, in interactive machine learning, of seeking relevant examples for the efficient training of a classifier. When Amershi et al. (2009) discuss how one might seek examples providing greatest information gain for the classifier, what they are really doing is building a partial command metamodel; a full metamodel would allow generative dialogue – their software would not only be able to identify examples from the existing corpus but also generate examples which satisfy perfectly the classifier's information need (provided that the human or other oracle who will label these examples can actually do a good job (Baum & Lang, 1992)). Groce et al. (2014) approach this from the perspective of end-user classifier testing, and show various strategies for selecting a testbed of evaluation examples. A method of eliciting examples is technically isomorphic to a command metamodel, since any such method must be able to define and identify deficiencies in the machine's training.

While algorithmic notions of "confidence" and "command" are not completely novel, they are usually not considered for the benefit of an advanced dialogue between human and model. The notion of "confidence", which is most mature, simply quantifies to human minds the quality of the prediction, but does not always suggest a further course of action. The notion of "command", in the case of reinforcement learning, is internal to the intelligent agent and embedded in its data structures, and not amenable to presentation or interpretation.

3.3 Complexity

The notion of complexity is the least discussed, and perhaps most interesting. How can exactly the same model produce more or less complex results? Consider the case of a neural network. It can be argued that when an input highly activates many of the nodes in a neural network, the decision making process is more complex than one which involves fewer nodes. This, despite the fact that the model structure is identical, with identical edge weights. It is analogous to the difference between mentally computing 199+101 and 364+487. One can follow an identical arithmetic "algorithm," and be equally confident in both answers, but one of these instances appears to be more complex than the other. A lot of what we think of as "complex" behaviour arises not out of a complex algorithm, but rather complex inputs. Deep Blue may have astonished with its famous defeat of Kasparov in 1997, but it did so not because it was following a complex algorithm; far from it. It did so because the input space and the domain carried with it considerable complexity. This idea is encapsulated in the allegory of Simon's ant

(Simon, 1996): observing an ant's convoluted, weaving path across a rocky beach, one might come away with the impression that the ant's behaviour is incredibly complex. However, the ant is only following extremely simple, local protocols for avoiding rocks and other obstacles as it attempts to achieve its general goal of returning home. The apparent complexity in its behaviour arises from the environment; it is not necessary to capture the complexity of the entire path in order to simulate an ant, but merely its localised obstacle-response protocols.

A "complexity" metamodel would capture this nuance: to produce a given output, has the model followed a parsimonious path from input to output, or grappled with a tortuous path through the rocks on a beach? This has historically been a fiendishly difficult problem for builders of practical machine learning systems, especially when attempting to generalise from small datasets. For example, when Cooper et al. (1997) were building a rule-based learning system to predict the likelihood of death from pneumonia in order to advise clinical treatment decisions, they discovered that the system was exploiting an artefact in the training data to make its predictions, namely that if the patient was an asthmatic, the model would actually predict a higher likelihood of survival! This absurd, medically demonstrable falsehood, was attributable to the fact that the model was trained on treatment records where asthmatics were given much more aggressive treatment in order to compensate for the poor status of their respiratory system, since they were known to have a greater risk of death. As a consequence, they had a better survival rate than non-asthmatics, and the model had picked up on this. Because of the complexity of the rule-based learning model being employed, it was impossible to guarantee that there were no such other inconsistencies in the model. This led to the model being dropped, and a much simpler class of "intelligible" models being adopted (Lou et al., 2012, 2013) despite having lower predictive power. This turned out to be a wise decision, as the model was subsequently found to be exploiting other similar false correlations. Similarly, researchers attempting to build a computer vision system for quantifying multiple sclerosis progression based on depth videos (Kontschieder et al., 2014) found that the system was exploiting patients' facial features in order to "remember" their training labels, so as to cheat the leave-one-out cross validation being used for evaluation.

Table 2 presents some suggestions for how complexity may be computed for popular machine learning techniques. Note that these are deliberately underspecified, and serve only to further illustrate what aspect of machine learning models is captured using the "complexity" notion.

Table 2. Practical complexity metamodel suggestions

Model	**Suggested calculations of complexity**
k-NN	The complexity of a given prediction can be measured as the variance of the distances for the k nearest neighbours. A larger variance can be interpreted as a more complex decision.
Neural Network	Setting a threshold t above which we consider a neuron to be "activated" (e.g., for a sigmoid activation function we might set $t = 0.9$), we can define the notion of "t-complexity", where the t-complexity of a neural network prediction is the fraction of the nodes in the network which are activated to level t or above.
Decision Tree	The tree-depth of a prediction provides a simple measurement for the complexity of a decision. For a more complex alternative, we might set a threshold i at which we consider the "majority" of the information gain to have been achieved, and define the notion of "i-complexity", where the i-complexity of a decision tree prediction is the tree-depth at which the cumulative information gain exceeds i en-route from the root to the outputted leaf node.
Naïve Bayes	Each classification decision in a Naïve Bayes classifier involves summing log probabilities of individual features given a class. The complexity of a Naïve Bayes classification decision can be measured as the variance of the log probabilities; a greater variance can be interpreted as a more complex decision.
Hidden Markov Model	Each of the primary HMM tasks will have different models of complexity. Intuition would suggest that a "simple" decision would be robust to small perturbations of the priors, transition function, and length of sequence that the algorithm is given to operate upon. Thus, if we define a threshold p on any of these quantities, then an HMM decision can be said to be p-simple if its output is robust to perturbations of magnitude less than p in its priors/transition function/sequence length.

It is important to note that the confidence and complexity measures are both always computed with respect to a given prediction. That is, whenever the model is used to make a prediction or classification, there is an associated value of confidence and complexity unique to that run of the algorithm. In contrast, the "command" metamodel refers to the current state of the algorithm's knowledge, not necessarily dependent on a single output.

3.4 Substituting metamodels as explanatory metaphors

We can take the technique of metamodelling for the facilitation of dialogue one step further, and achieve some interesting things, if we relax the accuracy constraint. That is, what if our metamodels don't strictly model what they're supposed to, but still provide plausible representations of that model's confidence, command, and complexity? This would be extremely useful in a case where our machine learning model is impossible to metamodel; we can nonetheless perform *metamodel substitution* to provide dialogue. Perhaps the decisions of a deep neural network are impossible to easily and correctly explain to an end-user, but if we present explanations as though the system is performing case-based reasoning (previously shown to be an intuitive approach (Sarkar et al., 2014)), then that may suffice. Thus, one metamodel can be used as a metaphor for another.

3.5 Metadialogue and metainteraction

The models of confidence, command, and complexity are repeatedly referred to as *meta*-models. I use this in the sense of "about," as in *metadata* (data about data) and *metacognition* (cognition about cognition), and so forth. At this level, the word "metamodel" simply refers to the fact that these are models about other (statistical and machine learning) models. However, there is room to discuss the treatment of the "meta-" prefix in the sense of "above," denoting a higher layer of abstraction, as in *metaphysics*, or perhaps *metatheory*. The primary object of study here is the interaction, or dialogue, between user and program, and not the models themselves. While we deal with the three metamodels as descriptors of the machine cognition, they could equally be descriptors of the interaction. For instance, while "complexity" is described here as a property of a statistical model, it could also be a property of the interaction itself, and this complexity may well be more noteworthy. This appears to bear greater relation to the problem of cognitive dimensions (Green & Petre, 1996), since it relates to the user experience of information structures as borne out through its visual and notational externalisations. Thus, it is quite possible that the brand of interaction we are considering here is more suitably called *meta*-interaction, or metadialogue. While a thorough treatment of this terminology is beyond the scope of the current discussion, a more detailed investigation in this direction would be an interesting subject of future study.

4 Analysis and applications

In this section I discuss how some examples of interactive machine learning systems are already benefiting from metamodel implementations, and can be usefully augmented by considering additional metacognitive models, or by newly considering their existing implementations.

4.1 Image segmentation

The *Crayons* application due to Fails and Olsen (Fails & Olsen, 2003) is a classic example of interactive machine learning. By "painting" positive and negative examples onto an image, the user can build a classifier which is able to segment areas of an image into two classes, e.g., a classifier which can classify human skin from non-skin objects in an image. It provides direct visual feedback on the image itself, by respectively darkening or lightening the negatively and positively classified images.

If a confidence metamodel was implemented, then instead of a standard intensity of darkening or lightening, the image could be overlaid with a colour whose intensity corresponded to the confidence with which pixels were classified as belonging into one class or another, as in Figure 4. This would further help the user refine their classifier, as it would be possible to identify regions which, while correctly classified, only just cross the decision boundary and thus have low confidence.

4.2 Email classification

Kulesza et al. have pursued a line of work which has investigated how one might assist users to debug rules learnt by a naïve Bayes classifier to categorise emails in various user-generated categories (Kulesza et al., 2009, 2011, 2015). They present *EluciDebug*, a visual tool for providing explanations of the naïve Bayes classifier's classification decision with respect to a given email. In doing so, they build an explicit metamodel of confidence, which can be used to sort emails and focus the user's attention on emails which may have been misclassified. They build an explicit metamodel of complexity, wherein the entire set of weights used to make the decision can be inspected through a series of bars, and thus it is apparent whether the classificiation was straightforward (dominated by a few clear high weights) or complex (wide distribution of potentially conflicting weights).

They also approach the development of a command metamodel; they use the sizes of different folders (which represent different classes) to explain the machine's prior beliefs regarding the likelihood of an unknown message belonging to any given class. This approaches a command metamodel since it alludes to the distribution of training examples the machine has thus encountered. However, it does not situate these examples in the input

Fig. 4. On the left: a simplified representation of the original *Crayons* interface, which shows classification of an image based on a binary threshold. On the right, a confidence-based visualisation which exposes how the classifier may still be uncertain about the fingers, and thus additional annotation may be beneficial.

domain. It is possible to envision a visualisation of all training examples, along with their text, projected from the high-dimensional space in which they reside onto a 2D manifold, such that deficiencies in the algorithm's experience can be identified. This is the precise approach taken by Amershi et al. (2009) for interactive image classification. A full command metamodel, defined with respect to the input space (finite-length finite-dictionary word vectors), would also enable the effective eliciting of appropriate training examples. This could take the form of either identifying emails which would greatly improve the overall confidence of the classifier if a label was obtained for them, or could extend to the artificial synthesis of an email whose label would satisfy the classifier's information needs.

4.3 Concept evolution in images

The *CueFlik* application due to Fogarty et al. (2008) presented a visual, programming-by-example method for designing classification rules to sort images in a database into different categories. Kulesza et al. (2014) expand upon this by acknowledging that users may not initially, or ever, have well-defined mental concept models (a key characteristic of the new programming paradigm discussed in §1.1), and so provide an interactive experience whereby the user is walked through a sequence of images which can be selected as belonging to a suggested class, or not. Automatic summaries of categories are generated to help the user remember what was distinctive about a particular category. Similar images from the corpus are displayed to assist the user in deciding whether creating a new category is warranted. Thus, the user can simultaneously refine their understanding, as well as the machine's understanding, of the categories they are creating.

Kulesza et al. provide suggestions for classes based on a recommendation system-like algorithm which compares the similarity of the image currently being classified to already-categorised images. Currently, the only feedback presented is in the form of a yellow star icon being placed next to the category the algorithm thinks is most appropriate. Using a confidence metamodel, one can envision an interface where different category labels are ranked, sized, or coloured according to the machine's confidence. This would help users identify categories which are potentially only weakly described by the training data.

A metamodel for complexity, driven by simplified representations of the input space, would potentially alert users to trivial simplifications being exploited by the algorithm, as in the examples presented in §3.3. One example of such a simplification is as follows: while categorising images of dogs and cats, it is possible that since most pictures of dogs are taken outdoors on green lawns, and most pictures of cats are taken indoors, what one is actually training is a classifier which detects the colour green. A complexity metamodel would be able to highlight how many, or which of the input image features are being used to make a decision, enabling the user to decide when to enrich the dataset or when to prune the feature space to prevent oversimplifications of the domain.

4.4 Analytical data modelling in spreadsheets

In Teach & Try (Sarkar et al., 2014), the user follows a two-step process to perform interactive machine learning in spreadsheets. The user first selects rows in which they have high confidence, and marks them using the "Teach" button. Next, the user selects rows to which they wish to apply the model, either in the form of populating empty cells with the model's predictions, or by evaluating the cells' current contents against the model's expectations. Pressing the "Try" button applies the model.

While fairly simplistic, we were able to show that the experience of interacting with the software led users to gain some appreciation of statistical procedures. During a post-experiment interview, participants were asked questions such as *how might the computer be doing this?*, and *why might the computer make a mistake?* It is important to note that none of our participants had any formal training in statistics or computing. Nonetheless, participants were able to informally articulate several potential algorithms (e.g., nearest-neighbours, case-based reasoning, and linear regression), as well as well-known issues with statistical modelling (e.g., insufficient data, insufficient dimensions, outliers, noise, etc.).

With a confidence metamodel, Teach and Try would enable users to critically evaluate its predictions. When a large number of predictions has been made, the confidence metamodel provides a heuristic with which the user can assess its performance; the user can choose to prioritise examining and correcting low-confidence predictions. With a command metamodel, it would be able to show users how the 'taught' rows are spread across the input domain, it would potentially be able to highlight areas of the data where receiving a user label would be beneficial, and potentially synthesise examples to be labelled.

Here, a complexity metamodel would again help users identify potential simplifications of the domain that the algorithm might be exploiting in order to perform its predictions, such as the pneumonia prediction and multiple sclerosis diagnostics examples given in §3.3. Another example might be as follows: a spreadsheet containing patient data, where each row represents a patient and each column represents various attributes of the patient (e.g., age, blood type, results of various diagnostic tests), may also have within it a 'date' field, representing the date that patient's entry was recorded. Prior to a certain date, only patients with a certain severity of illness were recorded in this spreadsheet. When using this spreadsheet to help assess whether or not a new patient may have a severe illness, a visualisation of the complexity model (perhaps in the form of how much each column contributed) might reveal that a decision tree has decided that the 'date' field contains enough information to conclude whether or not a patient will have a severe illness, and thus predicts, incorrectly (but nonetheless confidently), that no new patients can possibly be severely ill. Spotting this simplification, the user can take corrective measures such as excluding the date field, or removing the old records.

4.5 Commercial applications

Recommender systems: a common problem with music recommender systems, such as the engines underlying Pandora or iTunes radio, is that for an accurate model of your preferences to be built, the system needs to observe many examples of your listening history. As a consequence, users of such systems typically abandon the service before an accurate model is built, leading some to seek fast-converging estimates for recommendation systems, with varying levels of success. For example, Herbrich et al. (2007) tackle the issue of effectively recommending opponents in multiplayer games. It is important that opponents are well-matched, otherwise the game is not fun to play for either party. It is also important that these recommendations converge quickly, and that it is not necessary for a player to play several mismatched games before the system is able to correctly estimate their skill.

None of these recommendation systems exposes the underlying uncertainty associated with each prediction; by showing how the confidence of the system improves over time, and how its command of the domain of your music preferences improves as it is exposed to new examples (i.e., visible indicators of progress and improvement), the user may be more sympathetic to the amount of time required to properly train such systems.

Intelligent home devices: devices in our homes are getting increasingly intelligent. For instance, the Nest thermostat[1] learns your usage patterns throughout the day and begins to adjust itself. Similarly, certain refrigerators on the market will detect when you are running low on a particular item and place an online order on your behalf. These devices may ostensibly be programmed through purpose-built Internet-of-Things languages, such as IFTTT,[2] however, the primary programming interfaces many of these devices will have is through direct interaction, so that these interfaces can learn over time. In these situations, it can be quite important for the system to be able to express parts of its cognition to the user.

Driverless vehicles: the prospect of an autonomous car navigating its passengers past a complex array of obstacles at great speed evokes a visceral fear and suspicion, despite the fact that a tireless, emotionless controller with nanosecond reaction times can be orders of magnitude safer than human driving. Part of the reason for this reaction is that their interfaces have thus far been presented as completely opaque; the AI is portrayed to be in

[1] `https://nest.com/thermostat/life-with-nest-thermostat/` (last accessed July 15, 2015)
[2] `https://ifttt.com/wtf` (last accessed July 15, 2015)

complete control and the passengers have no intervention in its decision making process. Through metamodels of confidence, a driverless car might be able to identify situations where it defers to the judgment of a human driver. Similarly, through metamodels of command, the car might be able to identify road and scenery types which it had not previously encountered, and alert the driver to this.

5 Conclusion

I have discussed how we are undergoing a paradigm shift in programming, where the dominant mode of programming has moved from one with well-definable mental models to one without. This is accompanied by a movement from a direct, explicit information channel (*the program*) to an indirect, implicit, meta-information channel (*about the program*). Previous work in explanatory debugging and interactive machine learning has shown several different items which may be present in these meta-information channels, elevating our interaction with programs to a status resembling dialogue. To this channel, I have proposed a fundamental addition: models of machine metacognition. Unlike previous frameworks, mine is grounded in the engineering requirements for providing such types of information for intelligent systems.

I have argued for the utility and primacy of three models of machine self-metacognition: confidence in a given output, command of the problem domain, and complexity of the decision making process in producing a given output. I have presented some concrete suggestions for how such metamodels might be computed for popular machine learning algorithms. I have suggested how *metamodel substitution* may allow us to explain complex algorithms using simpler ones as metaphors. I have postulated a link between metamodels and meta-interaction. Finally, using examples from the literature in interactive machine learning and end-user debugging, I have demonstrated how these metamodels can enrich man-machine dialogue.

Acknowledgements

Many thanks to Alan Blackwell for his guidance on writing this paper and his proofreading of drafts at various levels of completion. Thanks to Cecily Morrison for recommending some of the relevant literature. My PhD is funded through an industrial CASE studentship sponsored by BT Research and Technology, and also by a premium studentship from the University of Cambridge Computer Laboratory.

References

Amershi, S., Fogarty, J., Kapoor, A., & Tan, D. (2009). Overview based example selection in end user interactive concept learning. In *Proceedings of the 22nd annual ACM symposium on User interface software and technology* (pp. 247–256).

Amershi, S., Fogarty, J., Kapoor, A., & Tan, D. S. (2011). Effective end-user interaction with machine learning. In *AAAI.*

Baum, E. B., & Lang, K. (1992). Query learning can work poorly when a human oracle is used. In *International joint conference on neural networks* (Vol. 8).

Behrisch, M., Korkmaz, F., Shao, L., & Schreck, T. (2014). Feedback-driven interactive exploration of large multidimensional data supported by visual classifier. In *Visual Analytics Science and Technology (VAST), 2014 IEEE Conference on* (pp. 43–52).

Blackwell, A. F. (2015). Interacting with an inferred world. In *Submission under review for the Decennial Aarhus conference.*

Brown, E. T., Liu, J., Brodley, C. E., & Chang, R. (2012). Dis-function: Learning distance functions interactively. In *Visual Analytics Science and Technology (VAST), 2012 IEEE Conference on* (pp. 83–92).

Carlin, B. P., & Gelfand, A. E. (1990). Approaches for empirical bayes confidence intervals. *Journal of the American Statistical Association*, *85*(409), 105–114.

Chryssolouris, G., Lee, M., & Ramsey, A. (1996). Confidence interval prediction for neural network models. *Neural Networks, IEEE Transactions on*, *7*(1), 229–232.

Cohn, D. A., Ghahramani, Z., & Jordan, M. I. (1996). Active learning with statistical models. *Journal of artificial intelligence research.*

Cooper, G. F., Aliferis, C. F., Ambrosino, R., Aronis, J., Buchanan, B. G., Caruana, R., ... others (1997). An evaluation of machine-learning methods for predicting pneumonia mortality. *Artificial intelligence in medicine*, *9*(2), 107–138.

Eddy, S. R. (2004). What is a hidden markov model? *Nature biotechnology*, *22*(10), 1315–1316.

Fails, J. A., & Olsen, D. R. (2003). Interactive machine learning. *Proceedings of the 8th international conference on Intelligent user interfaces - IUI '03*, 39. doi: 10.1145/604050.604056

Fiebrink, R., Cook, P. R., & Trueman, D. (2011). Human model evaluation in interactive supervised learning. *Proceedings of the 2011 annual conference on Human factors in computing systems - CHI '11*, 147. doi: 10.1145/1978942.1978965

Fogarty, J., Tan, D., Kapoor, A., & Winder, S. (2008). Cueflik: interactive concept learning in image search. In *Proceedings of the sigchi conference on human factors in computing systems* (pp. 29–38).

Gittins, J., Glazebrook, K., & Weber, R. (2011). *Multi-armed bandit allocation indices.* John Wiley & Sons.

González-Rubio, J., Ortiz-Martínez, D., & Casacuberta, F. (2010). Balancing User Effort and Translation Error in Interactive Machine translation via confidence measures. *Proceedings of the ACL 2010 Conference Short Papers, Uppsala, Sweden*, *173*(July), 173–177.

Green, T. R. G., & Petre, M. (1996). Usability analysis of visual programming environments: a cognitive dimensions framework. *Journal of Visual Languages & Computing*, *7*(2), 131–174.

Groce, A., Kulesza, T., Zhang, C., Shamasunder, S., Burnett, M., Wong, W.-K., ... McIntosh, K. (2014). You Are the Only Possible Oracle: Effective Test Selection for End Users of Interactive Machine Learning Systems. *IEEE Transactions on Software Engineering*, *40*(3), 307–323. doi: 10.1109/TSE.2013.59

Hao, M. C., Dayal, U., Keim, D. A., Morent, D., & Schneidewind, J. (2007). Intelligent visual analytics queries. In *Visual Analytics Science and Technology (VAST), 2007 IEEE Symposium on* (pp. 91–98).

Herbrich, R., Minka, T., & Graepel, T. (2007). Trueskill^{TM}: A bayesian skill rating system. In B. Schölkopf, J. Platt, & T. Hoffman (Eds.), *Advances in neural information processing systems 19* (pp. 569–576). MIT Press. Retrieved from `http://papers.nips.cc/paper/3079-trueskilltm-a-bayesian-skill-rating-system.pdf`

Herlocker, J. L., Konstan, J. A., & Riedl, J. (2000). Explaining collaborative filtering recommendations. In *Proceedings of the 2000 ACM conference on Computer supported cooperative work* (pp. 241–250).

Horvitz, E. (1999). Principles of mixed-initiative user interfaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems the CHI is the limit - CHI '99* (pp. 159–166). New York, New York, USA: ACM Press. doi: 10.1145/302979.303030

Kalkanis, G. (1993). The application of confidence interval error analysis to the design of decision tree classifiers. *Pattern Recognition Letters*, *14*(5), 355–361.

Ko, A. J., & Myers, B. A. (2004). Designing the whyline: a debugging interface for asking questions about program behavior. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 151–158).

Ko, A. J., Myers, B. A., Rosson, M. B., Rothermel, G., Shaw, M., Wiedenbeck, S., ... Lieberman, H. (2011, April). The state of the art in end-user software engineering. *ACM Computing Surveys*, *43*(3), 1–44. doi: 10.1145/1922649.1922658

Kontschieder, P., Dorn, J. F., Morrison, C., Corish, R., Zikic, D., Sellen, A., ... others (2014). Quantifying progression of multiple sclerosis via classification of depth videos. In *Medical image computing and computer-assisted intervention–miccai 2014* (pp. 429–437). Springer.

Kulesza, T., Amershi, S., Caruana, R., Fisher, D., & Charles, D. (2014). Structured labeling for facilitating concept evolution in machine learning. In *Proceedings of the 32nd annual acm conference on human factors in computing systems* (pp. 3075–3084).

Kulesza, T., Burnett, M., Wong, W.-k., & Stumpf, S. (2015). Principles of Explanatory Debugging to Personalize Interactive Machine Learning. In *Proceedings of the 20th international conference on intelligent user interfaces - iui '15* (pp. 126–137). doi: 10.1145/2678025.2701399

Kulesza, T., Stumpf, S., Burnett, M., Yang, S., Kwan, I., & Wong, W.-K. (2013). Too much, too little, or just right? Ways explanations impact end users' mental models. In *Proceedings of IEEE Symposium on Visual Languages and Human-Centric Computing, VL/HCC* (pp. 3–10). doi: 10.1109/VLHCC.2013.6645235

Kulesza, T., Stumpf, S., Wong, W.-K., Burnett, M. M., Perona, S., Ko, A., & Oberst, I. (2011). Why-oriented end-user debugging of naive bayes text classification. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, *1*(1), 2.

Kulesza, T., Wong, W.-K., Stumpf, S., Perona, S., White, R., Burnett, M. M., ... Ko, A. J. (2009). Fixing the program my computer learned: Barriers for end users, challenges for the machine. In *Proceedings of the 14th international conference on intelligent user interfaces* (pp. 187–196).

Laird, N. M., & Louis, T. A. (1987). Empirical bayes confidence intervals based on bootstrap samples. *Journal of the American Statistical Association*, *82*(399), 739–750.

Lim, B., & Dey, A. (2009). Assessing demand for intelligibility in context-aware applications. *Proceedings of the 11th international conference on Ubiquitous computing*, 195. doi: 10.1145/1620545.1620576

Lou, Y., Caruana, R., & Gehrke, J. (2012). Intelligible models for classification and regression. In *Proceedings of the 18th acm sigkdd international conference on knowledge discovery and data mining* (pp. 150–158).

Lou, Y., Caruana, R., Gehrke, J., & Hooker, G. (2013). Accurate intelligible models with pairwise interactions. In *Proceedings of the 19th acm sigkdd international conference on knowledge discovery and data mining* (pp. 623–631).

McSherry, D. (2005). Explanation in recommender systems. *Artificial Intelligence Review*, *24*(2), 179–197.

Nguyen, A., Yosinski, J., & Clune, J. (2015). Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. *Computer Vision and Pattern Recognition (CVPR '15), IEEE*.

Pu, P., & Chen, L. (2006). Trust building with explanation interfaces. In *Proceedings of the 11th international conference on intelligent user interfaces* (pp. 93–100).

Rothermel, G., Li, L., DuPuis, C., & Burnett, M. (1998). What you see is what you test: A methodology for testing form-based visual programs. In *Proceedings of the 20th international conference on software engineering* (pp. 198–207). Washington, DC, USA: IEEE Computer Society.

Sarkar, A., Blackwell, A. F., Jamnik, M., & Spott, M. (2014, July). Teach and try: A simple interaction technique for exploratory data modelling by end users. In *Visual Languages and Human-Centric Computing (VL/HCC), 2014 IEEE Symposium on* (pp. 53–56). IEEE. doi: 10.1109/VLHCC.2014.6883022

Savitha, R., Suresh, S., & Sundararajan, N. (2012). Metacognitive learning in a fully complex-valued radial basis function neural network. *Neural Computation*, *24*(5), 1297–1328.

Settles, B. (2010). Active learning literature survey. *University of Wisconsin, Madison*, *52*(55-66), 11.

Simon, H. A. (1996). *The sciences of the artificial* (Vol. 136). MIT press.

Smith, S. J., Bourgoin, M. O., Sims, K., & Voorhees, H. L. (1994). Handwritten character classification using nearest neighbor in large databases. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, *16*(9), 915–919.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., & Fergus, R. (2013). Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*.

Tintarev, N., & Masthoff, J. (2007). A survey of explanations in recommender systems. In *Data Engineering Workshop, 2007 IEEE 23rd International Conference on* (pp. 801–810).

Ware, M., Frank, E., Holmes, G., Hall, M., & Witten, I. H. (2001). Interactive machine learning: letting users build classifiers. *International Journal of Human-Computer Studies*, *55*(3), 281–292.

Watkins, C. J. C. H. (1989). *Learning from delayed rewards*. Unpublished doctoral dissertation, University of Cambridge England.

Weintraub, M., Beaufays, F., Rivlin, Z., Konig, Y., & Stolcke, A. (1997). Neural-network based measures of confidence for word recognition. In *Acoustics, speech, and signal processing, ieee international conference on* (Vol. 2, pp. 887–887).

Wilson, A., Burnett, M., Beckwith, L., Granatir, O., Casburn, L., Cook, C., ... Rothermel, G. (2003). Harnessing curiosity to increase correctness in end-user programming. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 305–312).

Zhang, L., & Luh, P. B. (2005). Neural network-based market clearing price prediction and confidence interval estimation with an improved extended kalman filter method. *Power Systems, IEEE Transactions on*, *20*(1), 59–66.