SAGA-ML: An Active Learning System for Semi-Automated Gameplay Analysis

Finnegan Southey and Robert C. Holte
University of Alberta, Dept. of Computing Science

Introduction

We present SAGA-ML, an *active learning* system for blackbox software testing. In our approach, labelled examples are obtained by sampling from the space of initial system states and user actions, and then running the blackbox to obtain the labels. This training set is then used to learn a model of the system's behaviour. Once a model has been learned, it is used to determine where further sampling should occur, increasing the training set for the next iteration of model learning. This intelligent sampling strategy attempts to allocate limited testing resources effectively.

Contemporary commercial video games are an example of very complex software systems that have tight development cycles and high reliability standards because aftermarket patching is not always an option. Furthermore, beyond the standard software correctness requirements, they have a more nebulous goal of "enjoyability" that is essentially impossible to specify or measure exactly. It is up to the designer to decide whether the game's behaviour is appropriate and the data collected can only assist them in the evaluation. In this context, machine learning serves a dual role where the learned model is used to summarize the game's behaviour for the developer, as well as directing the sampling of additional points.

We demonstrate our analysis tool, SAGA-ML (semi-automated gameplay analysis by machine learning), which is game independent and treats the game engine as a black box, and the SoccerViz visualization tool, a game-specific component that displays the learned models in a more comprehensible format. SAGA-ML has correctly discovered so-called "sweet spots" in the game (situations where it is too easy to score) and we demonstrate how these can be easily detected and explored by the developer using the SoccerViz tool. Here we will give a brief overview of the gameplay analysis task, the SAGA-ML architecture, and the SoccerViz tool. More detailed descriptions are available of our work on the gameplay analysis problem (Southey *et al.* 2005) and active learning as a software testing methodology (Xiao *et al.* 2005).

The Gameplay Analysis Task

While we believe the active learning framework has much to offer software testing in general, our demonstration is specific to the task of *gameplay analysis* for commercial computer games, a problem which offers a unique set of challenges over the typical software testing requirements. It is difficult to characterize the gameplay analysis task precisely. This is largely due to the fact that it inevitably involves some human judgement. "Enjoyability" is essentially impossible to quantify. An integral but complex part of enjoyability is the gameplay offered and an important aspect of good gameplay is appropriate difficulty. A game that is too hard is frustrating, while too little challenge can be boring.

In the gameplay analysis task, the developer selects some *metrics* for evaluating gameplay (e.g. probability of winning, average time to win/lose, average resources consumed to win, etc.). The scope of the analysis is necessarily limited to a small part of the game, since even these small pieces may be expensive to evaluate.

We use sampling and machine learning to obtain a model of the game's behaviour, offering a summary for the developer and also predictions for unsampled regions of the space. It is up to the developer to examine this information and decide whether the behaviour is acceptable or whether changes are required. This interactive approach leads us to call the process *semi-automated* and is why we emphasize the role of the learned model as a basis for visualization.

Electronic Arts' FIFA Soccer

We have used SAGA-ML to analyze scenarios in the Electronic Arts (EA) soccer game, FIFA Soccer. Games are generally too complex to be analyzed in their entirety. It makes sense to analyze certain specific scenarios, especially when the game itself distinguishes between scenarios (e.g. different levels, minigames, and control modes). We have explored a variety of scenarios in FIFA but only discuss the *shooter-goalie* scenario here.

In the shooter-goalie scenario, shown in Figure 1, the player controls a shooter placed somewhere near the goal and shooting into it, with the goalie placed to defend the goal. All other players are absent from the scenario. This tests the ability of the player to shoot and the goalie to block shots on goal, a critical part of the game.

Figure 1: The Shooter-Goalie scenario in FIFA'99.

Semi-Automated Gameplay Analysis

The overall architecture of the SAGA-ML approach is shown in Figure 2. The *game engine* is treated as a black box and SAGA-ML interacts with it through an *abstraction* layer. This layer is game-specific and translates game-specific data and function calls to an abstract state format used by SAGA-ML. The *sampler* component uses the abstraction layer to evaluate situations by running the game with an initial state and a sequence of actions, and then observing the outcome. The *learner* uses the data gathered by the sampler to construct a concise *model* (or *summary*) of the game's behaviour. The learner may then request more samples to refine its model. Together the sampler and learner form the active learning part of the system. Finally, the learned model is passed to the game-specific *visualizer* for the designer to evaluate.

In the current realization of SAGA-ML, we use the decision tree learner C4.5 (Quinlan 1994) to learn a set of IF-THEN style rules from the blackbox samples. These rules make predictions for different regions of the state-action space. For example, such a rule for the shooter-goalie scenario might predict "IF the shooter is within 5 metres of the goalie AND the angle between shooter and goalie is between $30°$ and $40°$ AND the goalie is within 1 metre of the goal's centre THEN the probability of scoring is greater than 70%". These rules can be visualized by the developer using SoccerViz. They are also used to decide where the next iteration of sampling should occur. This is known as *active learning*. SAGA-ML currently implements a variety of active learners including *uncertainty sampling*, *Query by Boosting* and *Query by Bagging*, *Bootstrap-LV*, and a new method we developed ourselves, *decision boundary refinement sampling* (see (Xiao *et al.* 2005) for details and references).

Visualization

Visualization is an important aspect of our framework, and one that is best handled by game developers who have considerable graphics, user-interface, and gameplay design experience. Nevertheless, we have developed a tool to demonstrate how SAGA-ML usage might feature in real game development. The SoccerViz visualization tool is shown in Figure 3. This tool displays the rules generated by the learning as regions on a 2-D soccer field and allows the user to examine the samples supporting the rule. These samples can also be exported to the FIFA engine (see Figure 1) in order to fully examine the game's behaviour for that sample. This visualization is clearly specialized for soccer, and some customization was required for the different scenarios. We believe that, in industry practice, much of the visualization could be incorporated into the real game engine, or into its associated design tools, in a cost-effective manner.

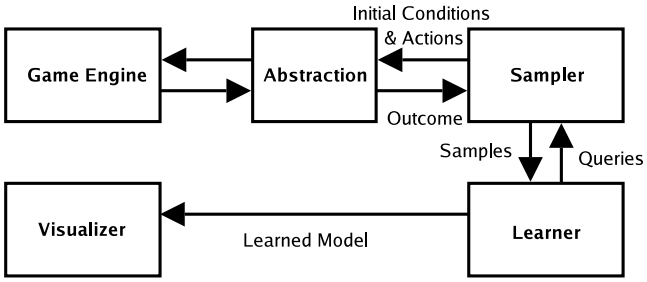

Figure 2: Architecture

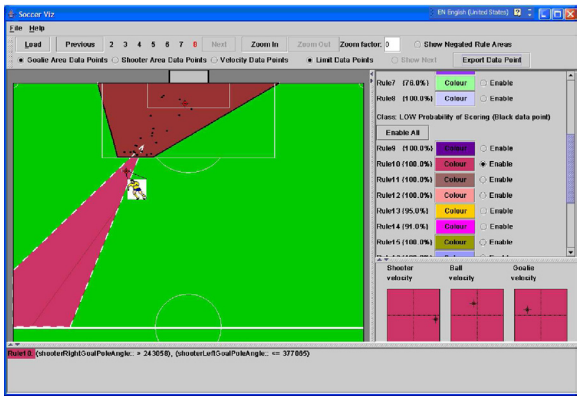

Figure 3: SoccerViz display of a rule predicting low scoring probability. Shading near the goal shows goalie positions covered by the rule. The triangle in the field shows covered shooter positions. Black dots show sampled non-scoring goalie positions. One particular shooter/goalie position sample is shown and was exported to the game engine to obtain Figure [fig:Shooter-Goalie-Engine].

Acknowledgements

Thanks to the Natural Sciences and Engineering Research Council of Canada, IRIS, and the Alberta Ingenuity Centre for Machine Learning for their financial support, and Electronic Arts for the FIFA source and continuing feedback.

References

Quinlan, J. R. 1994. *C4.5 : Programs for Machine Learning*. Morgan Kaufmann.

Southey, F.; Holte, R. C.; Xiao, G.; Trommelen, M.; and Buchanan, J. 2005. Machine Learning for Semi-Automated Gameplay Analysis. In *Proceedings of the 2005 Game Developers Conference (GDC-2005)*.

Xiao, G.; Southey, F.; Holte, R. C.; and Wilkinson, D. 2005. Software Testing by Active Learning for Commercial Games. In *Proceedings of the 20th National Conference on Artificial Intelligence (AAAI-05)*.