Classification on Graphs for Dynamic Difficulty Adjustment: More Questions Than Answers

Olana Missura OLANA.MISSURA@IAIS.FRAUNHOFER.DE
Thomas Gärtner THOMAS.GAERTNER@IAIS.FRAUNHOFER.DE
Fraunhofer Institute for Intelligent Analysis and Information Systems IAIS
Schloss Birlinghoven, D-53754 Sankt Augustin, Germany

Abstract

Motivated by a problem of dynamic difficulty adjustment in computer games, we propose a new learning setting. We argue that video games require a mechanism for dynamic difficulty adjustment. Such mechanism can be formulated as a classification problem on a graph that combines the properties of online and active learning. The main contribution of this paper is the clarification of this novel learning setting and a summary of open questions.

1. Introduction

A game and its player are two interacting entities. A typical player plays to have fun, while a typical game wants its players to have fun.What constitutes the *fun* when playing a game? One theory is that our brains are physiologically driven by a desire to learn something new: new skills, new patterns, new ideas (Biederman & Vessel, 2006). We have an instinct to play because during our evolution as a species playing generally provided a safe way of learning new things that were potentially beneficial for our life. Daniel Cook (Cook, 2007) created a psychological model of a player as an entity that is driven to learn new skills that are high in perceived value. This drive works because we are rewarded for each new mastered skill or gained knowledge: The moment of mastery provides us with the feeling of joy. The games create additional rewards for their players such as new items available, new areas to explore. At the same time there are new challenges to overcome, new goals to achieve, and new skills to learn, which creates a loop of learning-mastery-reward and keeps the player involved and engaged.

Thus, an important ingredient of the games that are fun to play is providing the players with the challenges corresponding to their skills. It appears that an inherent property of any challenge (and of the learning required to master it) is its difficulty level. Here the difficulty is a subjective factor that stems from the interaction between the player and the challenge. The perceived difficulty is also not a static property: It changes with the time that the player spends learning a skill.

The dependency between the perceived difficulty and player's skills is bidirectional: The ability to learn the skill and the speed of the learning process are also controlled by how difficult the player perceives the task. If the bar is set too high and the task appears too difficult, the player will end up frustrated and will give up on the process in favour of something more rewarding. Then again if the challenge turns out to be too easy (meaning that the player already possesses the skill necessary to deal with it) then there is no learning involved, which makes the game appear boring.

It becomes obvious that the game should provide the challenges for the player of the "right" difficulty level: The one that stimulates the learning without pushing the players too far or not enough. Ideally then, the difficulty of any particular instance of the game should be determined by who is playing it at this moment. In other words, the game should possess an ability to change the difficulty of its challenges on the fly, in an online fashion.

2. General Setting and Related Work

In the games existing today we can see two general approaches to the question of difficulty adjustment. The traditional way is to provide a player with a way to set up the difficulty level for herself. Unfortunately, this method is rarely satisfactory. For game developers it is not an easy task to map a complex gameworld into a single parameter. When constructed, such a mapping requires additional extensive testing, creating time and money costs. Additionally, in general games require several different skills to play

Presented at *ILP-MLG-SRL*, Leuven, Belgium, 2009.

them. The necessity of going back and forth between the gameplay and the settings when the tasks become too difficult or too easy disrupts the flow component of the game.

An alternative way is to implement a mechanism for dynamic difficult adjustment (DDA). There exist a few games with a well designed DDA mechanism (Capcom's *Resident Evil 4* being probably most known example), but all of them employ heuristics and as such suffer from the typical disadvantages of heuristics (being not transferable easily to other games, requiring extensive testing, etc). What we would like to have instead of heuristics is a universal mechanism for DDA: An online algorithm that takes as an input (game-specific) ways to modify difficulty and the current player's in-game history (actions, performance, reactions, ...) and produces as an output an appropriate difficulty modification.

Both artificial intelligence researchers and the game developers community display an interest in the problem of automatic difficulty scaling. Different approaches can be seen in the work of R. Hunicke and V. Chapman (Hunicke & Chapman, 2004), R. Herbich and T. Graepel (Herbich et al., 2006), Danzi et al (Danzi et al., 2003), and others. As can be seen from these examples the problem of dynamic difficulty adjustment in video games was attacked from different angles, but a unifying approach is still missing.

Let us reiterate that as the perceived difficulty and the preferred difficulty are subjective parameters, the DDA algorithm should be able to choose the "right" difficulty level in a comparatively short time for any particular player. Since both these parameters change with time, the algorithm should keep adjusting the difficulty in the right manner as the player progresses through the game. Note also that in the general case the ways to modify difficulty don't have to be labelled (as in "this action increases the difficulty" and "this one decreases it"), since even that can depend on the player and her current state: For someone with a grenade launcher a turret presents a different level of challenge than for someone armed with a crowbar only.

In the following sections we present a formalisation of the DDA problem and list the directions for the future work.

3. Formalisation

Let us formalise the problem in the following way. Instead of a property representing a difficulty level, we assume the game has a set of features, all of which can influence the perceived difficulty of a particular player. Each game state v is a point in the game's feature space. As described above, the perceived difficulty is a subjective value that directly depends on the game's player at this specific moment. Therefore, we say that we are given in some way a player instance p_t at time t.

The player instance p_t defines a partial order on the set of game states V, $\succeq_{p_t}$, such that $v_1 \succeq_{p_t} v_2$ if and only if the player p at time t perceives the state v_1 as more or equally difficult as the state v_2. Furthermore, the player instance p_t defines a labelling on V, $y_{p_t} : V \to \{+1, 0, -1\}$, where the label depends on the player's perceived difficulty of the state v:

$$y_{p_t}(v) = \begin{cases} +1 & \text{too difficult,} \\ 0 & \text{engaging,} \\ -1 & \text{too easy.} \end{cases}$$

Note that the labelling y_{p_t} is monotone with regard to the partial order $\succeq_{p_t}$, that is $y_{p_t}(v_1) \geq y_{p_t}(v_2)$ if and only if $v_1 \succeq_{p_t} v_2$.

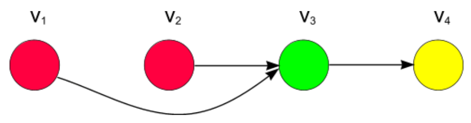

Figure 1. An example of the graph G_{p_t} and the labelling y_{p_t} on it induced by the player instance p_t (too difficult states are coloured red, engaging ones green, and too easy ones yellow). From the graph and the labelling we can see that this player considers the game states v_1 too difficult,that she perceives the game state v_4 as being easier than v_3, etc.

One instance of a game consists of a subset of the game states that the player and the game traverse together. Note, that due to the structure of any particular game there exist constraints regarding the allowed states transitions. To formalise these constraints we introduce two sets of edges on V: E and F. The edges in E represent the possible transitions of the player, while the edges in F are the transitions allowed for the game itself. For example, in a computer role-playing game (CRPG) among the edges in E are the player's movements or his decisions on which weapon to use, while among the edges in F are the game's decisions on how many monsters to place in the player's vicinity.

To summarise, we define a game graph G_{p_t} as a five-tuple $(V, \succeq_{p_t}, E, F, y_{p_t})$. A game instance is a trajectory on G_{p_t}, $T = (u_1, v_1, u_2, v_2, \ldots, u_t, v_t, \ldots)$, where $(u_i, v_i) \in F$ and $(v_i, u_{i+1}) \in E$. That is, we assume an iterative nature of the game instance: After each player's transition the game has an opportunity to modify something, and vice versa. To account for the possibility of nothing happening any of the edges in T can be self-loops. The task of keeping the player engaged by providing her with adequately difficult challenges turns into (1) the question of finding the subset of vertices $V_0 = \{v \in V | y_{p_t}(v) = 0\}$ and (2) the question of how to place a player in a challenging state. In other words, we face the tradeoff between exploration (for 1) and exploitation (for 2). The goal of the learning algorithm is to find V_0 while minimising $\sum_{v_i \in T} |y_{p_t}(v_i)|$.

On the one hand, we face an online learning problem because while playing the game the player is moving along a sequence of game states, effectively choosing the learning instances. Plus, we want to minimize the amount of mistakes that the algorithm makes while predicting the labels. On the other hand, since the game has the power to change the game states to find the best suitable ones for a given player, it should utilise this power to accelerate the learning algorithm by choosing the game states that will provide the most information to the algorithm, which puts us into an active learning scenario.

4. Research Directions

Several questions need to be taken into consideration regarding the formalisation of the DDA task described above.

Partial Order. Recall that the graph G_{p_t} is defined by the partial order $\succeq_{p_t}$ induced by a particular player p at the time t. Unfortunately, this partial order is not given to us and the graph is not known. We suggest that a combination of domain knowledge and user experiments can be used to overcome this problem. Facts from the domain knowledge can be utilised to create the edges in G_{p_t} that are valid universally, for all players. User experiments allow for collecting explicit feedback regarding $\succeq_{p_t}$. Clustering the resulting partial orders can lead us to the types of the players. When encountering a new player, it would be enough to estimate which cluster she belongs to to get access to her particular $\succeq_{p_t}$.

True Labels. It is important to note that in contrast to the standard online or active learning setting, neither the player nor the game provide the learning algorithm with the true labels of the game states $y_{p_t}(v)$. Therefore, a mechanism needs to be developed to infer a true label from the available observations.

Concept drift. Due to the nature of the DDA task, the player's concept y_{p_t} changes with time. Luckily for us it does not change arbitrarily. Rather, we can safely assume that, while playing the game, the player's skill with it increases, and the game states that appeared too difficult for her turn into engaging ones, while the once engaging states move into the 'too easy' class. In other words if we denote by $p_{t'}$ the instance of the player p at the time t', where $t' > t$, then for any game state v $y_{p_{t'}}(v) \leq y_{p_t}(v)$. The learning algorithm should be able to deal with this drift. In other words, it is not enough to classify the states once and for all, but it is necessary to query the already classified states now and again. The question is, which states in particular and how often?

Objective function. The objective function described above can potentially lead to the infinite costs. Consider Figure 2: The player starts in the state v_0 and the learning algorithm wants to move her to the state v_2. After the first transition the player finds herself in the state v_1 and moves back to v_0. Now every time the algorithm moves her to v_1, she goes straight back to v_0, producing infinite costs. One possibility to work around this problem is to make some assumptions for the player's behaviour that prevent the situation described above. For example, we can safely assume that the player is 'curious' meaning that she prefers to see and experience new game states rather than come back to the same old ones.

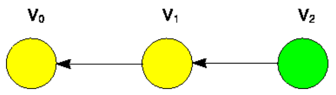

Figure 2. A set up for potentially infinite costs.

The main question is of course how to design a learning algorithm able to solve the DDA problem and what performance guarantees can such an algorithm have. To this end we propose an approach along the lines of the algorithm described in (Gärtner & Garriga, 2007). In essence, we suggest to perform a binary search on a path in the minimum path cover, until we find a game state that is satisfactory. In this way we can provide the guarantees on the number of queries the algorithm needs to perform to find an engaging state in a simplified setting. To address the DDA problem in its general setting we will investigate the questions listed above in our future work.

Acknowledgments

We would like to thank the anonymous reviewers for their helpful and insightful comments.

References

Biederman, I., & Vessel, E. (2006). Perceptual pleasure and the brain. *American Scientist*, *94*.

Cook, D. (2007). The chemistry of game design. World Wide Web electronic publication.

Danzi, G., Santana, A. H. P., Furtado, A. W. B., Gouveia, A. R., Leitão, A., & Ramalho, G. L. (2003). Online adaptation of computer games agents: A reinforcement learning approach. *II Workshop de Jogos e Entretenimento Digital*, 105–112.

Gärtner, T., & Garriga, G. C. (2007). The cost of learning directed cuts. *Proceedings of the 5th International Workshop on Mining and Learning with Graphs*.

Herbrich, R., Minka, T., & Graepel, T. (2006). Trueskill$^{\text{tm}}$: A bayesian skill rating system. *NIPS* (pp. 569–576).

Hunicke, R., & Chapman, V. (2004). AI for dynamic difficulty adjustment in games. *Proceedings of the Challenges in Game AI Workshop, Nineteenth National Conference on Artificial Intelligence*.