Data Stream Mining Algorithms for Building Decision Models in a Computer RolePlaying Game Simulation

Rosane M. M. Vallim
Instituto de Ciências Matemáticas
e de Computação - ICMC
Universidade de São Paulo

André C. P. L. F. Carvalho
Instituto de Ciências Matemáticas
e de Computação - ICMC
Universidade de São Paulo

João Gama
LIAAD, INESC-Porto
Universidade do Porto

Abstract

Computer games are attracting increasing interest in the Artificial Intelligence (AI) research community, mainly because games involve reasoning, planning and learning [Fürnkranz 2007]. One area of particular interest in the last years is the creation of adaptive game AI. Adaptive game AI is the implementation of AI in computer games that holds the ability to adapt to changing circumstances, i.e., to exhibit adaptive behavior during the play. This kind of adaptation can be created using Machine Learning techniques, such as neural networks, reinforcement learning and bioinspired methods. In order to learn online, a system needs to overcome the main difficulties imposed by games: processing time and memory requirements. Learning in a game needs to be fast and the memory available is usually not enough to store a large number of training examples to a traditional Machine Learning technique. In this context, methods for mining data streams seem to be a natural approach. Data streams are, by definition, sequences of training examples that arrive over time [Gama and Rodrigues 2009]. In the data stream scenario, algorithms are usually incremental and capable of adapting the decision model when a change in the distribution of the training examples is detected. One particularly interesting algorithm for mining data streams is the Very Fast Decision Tree (VFDT) [Domingos and Hulten 2000]. VFDTs are incremental decision trees designed specifically to meet the data stream problem requirements. In this paper, we analyse the use of VFDTs in the task of learning in a Computer RolePlaying Game context. First, we simulate data from manually designed tactics for a Computer RolePlaying Game, based on Spronck's static tactics [Spronck 2005], and test the suitability of VFDT to rapid learn these tactics. Afterwards, we conduct an experiment in order to simulate a data stream of examples where changes of tactics occur over time, and analyse how VFDT and some of its variations respond to these changes in the target concept.

Keywords:: Computer RolePlaying Games, Data Stream Mining, Player Modelling

Author's Contact:

{rvallim,andre}@icmc.usp.br
jgama@fep.up.pt

1 Introduction

In the recent years, the interest of game developers and researchers have shifted from realistic graphics to more believable AI. The reason is that over the last decades, computer games have become very realistic in terms of graphics and sound effects, leaving AI as an underexplored part of the development. Because of that, game AI is basically constructed using a variety of nonadaptive techniques, which has the well known disadvantage of being static [Bakkes et al. 2009]. In order to produce next level game AI, researchers have focused their attention in Machine Learning (ML) techniques, with the objective of producing adaptive game AI, i.e., the capability of adapting the behavior of AI controlled characters for better responding to changes in the game environment.

Several ML techniques have been applied to the games scenario. In spite of that, game developers are still concerned about introducing these techniques in a real game, mainly because the traditional ML algorithms usually need numerous trials to learn an effective model. Additionally, some of these algorithms rely on stochastic procedures, which can result in unpredictable, and also difficult to reproduce, behavior. As a result, the majority of ML algorithms and techniques used in games is applied offline, which is known as offline learning [Millington 2006]. Offline learning is employed before the game is released for sales with the objective of exploring and exposing possible design weaknesses and improving AI quality. Online learning, in the counter side, allows game AI to automatically detect and adapt to changing circumstances while the game is in progress, and even repair weaknesses that have been exploited by the human player [Spronck et al. 2003]. One of the main advantages of adaptive game AI is the possibility of adapting game AI to changing player tactics and styles.

A particular class of games where the incorporation of adaptive game AI is more complicated is probably the Computer RolePlaying Game (CRPG). This is due to the number of possible actions in these games that ranges from hundreds to even thousands in each turn. Because of this characteristic, CRPGs are most of the time coded using scripts, which are a set of rules written in a high level language that controls the behavior of the opponent characters. A relatively novel technique is Dynamic Scripting [Spronck 2005], a reinforcement learning technique that uses adaptive rulebases for the formation of scripts.

In this paper, the main objective is to explore the usage of data stream mining techniques in the context of game AI. Data stream mining is an area of research concerned with the development of learning algorithms for complex, dynamic environments where data flows in a continuous way. Moreover, these algorithms have to be able to work at non-stationary environments, i.e., environments where the distribution generating the training examples can change over time, also called concept changes. Because of this, algorithms for mining data streams have to continuously adapt their decision model to new examples. According to Domingos and Hulten [Domingos and Hulten 2000] some of the desirable properties of learning systems that mine data streams are: i) incrementality, ii) online learning, iii) constant time to process an example, using fixed memory, iv) single scan over the training data and v) taking concept changes into account. Because of these properties of data streams algorithms, we believe they are a natural approach for the problem of learning in games, and more specifically, learning combat tactics in CRPGs.

In this work, we study the VFDT algorithm, and some of its variations, for effectively learning different combat tactics in CRPGs. We simulated data using static tactics based on the ones present in the work of Spronck [Spronck 2005] in order to use as training examples for the algorithms analysed. We simulated data from two of the tactics, namely, the offensive tactic and the defensive tactic, for both types of characters (fighter and wizard). The objective was to analyse how fast the VFDT can learn these tactics.

We also experimented with a simulated stream of data where examples from both tactics are interchanged from time to time. The objective of this experiment is to demonstrate the usability of data stream mining algorithms designed to tackle concept change in the context of computer games.

The approach conducted in this paper is mainly motivated by a known research area in game playing called Player Modelling, where the goal is to improve the opponent player by allowing it to adapt to its opponent [Fürnkranz 2007]. More specifically, the approach presented here is inspired in behavioral cloning techniques, where artificial players's strength and credibility are increased by imitating the strategy of a human player.

This paper is organized as follows. In Section 2, a small discussion on opponent AI in CRPGs is described, focusing on adaptive techniques. Section 3 brings a brief introduction to data stream mining. Section 4 presents the original VFDT algorithm and the extensions used in this work. Next, in section 5, methods based on ensembles of VFDTs are presented, as they were used in the experiments. Section 6 presents the approach followed in this work. Section 7 brings the experiments conducted and their results. Finally, section 8 concludes the paper and gives some directions for future research.

2 Related Works

CRPGs are a type of game where the human player is represented in a virtual world by a character (or a group of characters) that goes on a quest that involves, among other things, solving puzzles, talking to the world's inhabitants and defeating enemies in combat. Combats in CRPGs include, at each combat round, a variety of possible choices available to the players, turning decision making in this kind of game a complex task of reasoning.

To deal with these complex games, designers usually employ scripts, which are lists of rules that are executed sequentially. This scripts are usually long and complex, and are based on knowledge domain. The major problem with scripts is their static nature. A list of rules models an opponent behavior, and this behavior will not change until the end of the game. Because in modern CRPGs scripts tend to be complex in order to model the desired behaviors, they are also much more likely to contain weaknesses. Once the human players discover one of this weaknesses, they will exploit it to defeat opponents designed to be tough. Besides, scripts are not adaptable and cannot deal with tactics that were not unforeseen by the designers [Spronck et al. 2003].

One major advance to the scripting technique is Dynamic Scripting (DS) [Spronck 2005]. DS is a reinforcement learning technique that uses different rulebases for each type of opponent in the game. The scripts of each opponent are created selecting rules from the corresponding rulebase. The main innovation of DS is that it can adapt its rulebases by assigning weights to the rules. By the moment a new opponent is created, the rules that will form the opponent's script are selected from the corresponding rulebase. The probability of a rule being selected is proportional to its weight. The adaptation of the rulebases are carried out by updating the weights of the rules, depending on their corresponding sucess or failure. Some works have extended the basic idea of DS, including extra steps [Szita et al. 2009].

In the work of Crocomo et al [Crocomo et al. 2007], a Genetic Algorithm (GA) is used to learn effective combat behavior in CRPGs. Each chromosome in the GA's population if formed by a set of rules, defining the behavior of a group of characters. These chromosomes are evaluated in the game environment, and a fitness is given to each one depending on their success on combats. A new population is generated, using crossover and mutation operators, and the process repeats until a stopping criteria is met.

3 Data Stream Mining

Data streams are sequences of examples that arrive at a high rate and that, frequently, can be read only once using a small amount of processing time and memory. Since it is not possible to store all the examples that arrive in a stream, learning algorithms designed to learn in this context analyse the training examples only once. Besides, it is necessary that this algorithms be capable of updating the decision model every time new examples become available.

In the context of data stream mining, we cannot assume that the examples are generated according to a stationary probability distribution. Since in these problems the distribution gererating the examples is non-stationary, the concept underlying the examples can change over time. Therefore, old information that determined the behabior of the decision model is no more appropriate to represent this probability distribution. This is referred in the literature as concept change [Gama and Rodrigues 2007; Gama et al. 2004].

In the data stream literature there are two main approaches to deal with concept changes. The first one updates the decision model in fixed intervals and does not take into account if changes have really happened. Many algorithms that fall into this approach use time windows, where the decision model is induced only with the examples contained in this window. The main concern with the use of time windows is the definition of the windows's size [Gama and Rodrigues 2007].

The second approach monitors some indicators in order to detect possible concept changes. If a change is detected, the algorithm takes appropriate actions to adjust the decision model according to the change. One possible action is to adjust the size of the time window used. The methods following this approach are often referred in the literature as drift detection methods.

Many methods for detecting and reacting to concept drift are reported in the literature. One of these methods is proposed by Gama et al [Gama et al. 2004] and is here referred as DDM (from Drift Detection Method). The idea behind DDM is to control the online error-rate of the algorithm. When a new training example is available, it is classified using the actual model and the error-rate of the model can be calculated. The authors define a warning level and a drift level. A new context is declared if, in a sequence of examples, the error-rate increases reaching the warning level at example k_w and the drift level at example k_d. The algorithm then learns a new model using only the examples since k_w. Another method, EDDM (Extended Drift Detection Method) [Baena-García et al. 2006] extends DDM. In EDDM, the estimated distribution of the distances between classification errors is used in order to improve the window resize method used in DDM.

4 Very Fast Decision Trees

VFDTs are incremental decision tree learners that build decision models from extremely large datasets, making it possible to directly mine data streams without ever needing to store the examples. These systems were designed to build potentially very complex trees with an acceptable computational cost.

As stated by Catlett [Catlett 1991], in order to find the best split attribute at a given tree's node it is only necessary to consider a subset of the training examples that reach this node. In this manner, VFDTs build decision tree models by recursively changing leaf nodes by decision nodes.

Each leaf in the tree stores the so-called sufficient statistics about attribute-values. These statistics are the ones necessary to calculate an heuristic evaluation function used to evaluate the merit of inserting a split decision based on attribute-values. Examples of heuristic functions are the Gini Index and the Information Gain. The labeled examples that come in the stream traverses the tree from the root to a leaf node, testing the appropriate split attributes in the way. When the example reaches a leaf node, all the sufficient statistics are updated. When a minimum number of examples visit a leaf each possible splitting test based on attributes-values is verified. If there is enough statistical support in favor of one splitting test over the others, then the leaf is transformed into a decision node and two new descendant empty leaves are created.

VFDTs use the Hoeffding bound to decide how many examples a leaf should observe before installing a decision node. Suppose n independent observations of a random variable r, whose range is R, were performed. The Hoeffding bound states that, with probability $1 - \delta$, the true average of r, denoted by $\overline{r}$, is in the interval $\overline{r} \pm \epsilon$, where $\epsilon = \sqrt{R^2 \frac{\ln(\frac{1}{\delta})}{2n}}$. Next, let H be the heuristic evaluation function of an attribute. For the Information Gain, the maximum value R of H is equal to $\log_2(\#classes)$. Let x_a be the attribute with highest value of H, x_b the attribute with second-highest value of H and $\Delta H = H(x_a) - H(x_b)$ the difference between the two best attributes. Then, if $\Delta H > \epsilon$ with n training examples observed at the leaf, the Hoeffding bound states that, with probability $1 - \delta$, x_a is really the attribute with highest value of the heuristic evaluation function. This means that the leaf node should be transformed into a decision node that splits on x_a.

VFDTs use a constant, τ, for run-off in the case where the Hoeffding bound would not decide for an attribute over the others. This situation can happen when two or more attributes have very similar values of H, even with a large number of examples observed. Then, if $\Delta H < \epsilon < \tau$, the leaf is transformed into a decision node and the split attribute is the one with highest value of H.

The basic VFDT algorithm has been extended to process continuous attributes, to allow the use of the Naive-Bayes classifier at the leaves and to detect and react to concept drift [Gama et al. 2006]. In this work, we consider the algorithm with the two first extensions and, therefore, not able to react to concept changes. This algorithm will be referred in the rest of this paper as VFDT-NB.

5 Ensembles of VFDTs

Ensemble methods consist of combinations of various decision models, which individual predictions are combined according to some strategy in order to form a unique final prediction. The individual predictions can be combined using some voting strategy. In general, ensembles of classifiers achieve a higher accuracy than a single classifier and are also simpler to scale and parallelize. Besides, ensembles can quickly adapt to changes by means of pruning parts of the ensemble with inferior performance. These characteristics make ensemble methods more suitable for solving non-stationary problems than conventional methods [Bifet et al. 2009].

Two ensemble methods using VFDTs were proposed by Bifet et al [Bifet et al. 2009]. The first method, Bagging Adaptive-Size Hoeffding Trees, consists of a bag of VFDT trees each one trained with a different sample of examples. Adaptive-Size Hoeffding Tree (ASHT) is a direct derivation of the VFDT, imposing a limited depth in the tree. The maximum number of decision nodes of the first tree is equal to 2. The maximum number of decision nodes of the n_{th} tree is two times the maximum number of nodes of the tree with index $n - 1$. Each tree has also an associated weight that is proportional to the inverse of the square of its error. If the number of decision nodes in a tree exceeds the maximum number of nodes allowed, then the algorithm deletes some nodes to reduce the size of the tree.

The second method uses Bagging VFDT in association with the ADWIN algorithm [Bifet and Gavaldà 2007]. The Bagging algorithm used is the one proposed by Oza and Russel [Oza and Russel 2001] that uses the Poisson distribution for sampling the examples used to train each tree. ADWIN is an algorithm for detecting changes and an estimator that maintains a window of variable size of the most recent examples. This window has the maximum possible size in order that the mean of the values inside the window has not changed. Each individual tree uses the ADWIN algorithm to detect changes in a data stream and to estimate the weight of the tree itself. In their strategy, when a change is detected, the worst classifier is removed from the ensemble and a new classifier is created.

Pfahringer et al [Pfahringer et al. 2007] proposed a method based on the Hoeffding Option Trees (HOT) [Kohavi and Kunz 1997]. Hoeffding Option Trees are VFDTs where some decision nodes contain several splitting tests based on different attributes. These nodes are designated by optional nodes. In optional nodes, each split test has an associated subtree. The interesting thing about the Hoeffding Option Trees is that these trees consist of several VFDTs in the same structure. In [Pfahringer et al. 2007], a variation of these trees is proposed, the so-called Adaptive Hoeffding Option Trees (ASHOT), in which each leaf stores an estimative of the actual error in order to calculate the weight of each leaf in the voting scheme. The weight of a node is proportional to the square of the inverse of its error.

6 VFDTs for Building Decision Models in a CRPG

In this work, we propose the use of VFDTs for building decision models from CRPG data. The two main objectives of this work are, first, analyse how fast a VFDT can learn static combat tactics. Second, show the potential of VFDTs to adapt the decision model to changing combat tactics.

For reaching the first objective, we analysed the VFDT-NB algorithm in simulated CRPG data. These data were created according to the static tactics presented in the work of Spronck. For such, two different tactics were used, offensive and defensive, for two different types of opponents, a fighter and a wizard.

The fighter's offensive tactic is represented by the following rules, where HP means health percentage:

- **if** $(HP < 50)$ **and** $(PotionHealing = yes)$

  then DrinkPotionH
- **if** $(HP \geq 50)$ **then** AttackClosestEnemy

The wizard's offensive tactic is formed by the rules:

- **if** $(HP < 50)$ **and** $(PotionHealing = yes)$

  then DrinkPotionH
- **if** $(HP \geq 50)$ **and** $(Level3SpellDG = yes)$

  and $(TypeClosestEnemy = wizard)$

  then CastLevel3DGSpellCenterEnemy
- **if** $(HP \geq 50)$ **and** $(Level2SpellDG = yes)$

  then CastLevel2DGSpellClosestEnemy
- **if** $(HP \geq 50)$ **and** $(Level1SpellDG = yes)$

  then CastLevel1DGSpellWeakestEnemy
- **if** $(HP \geq 50)$ **then** AttackClosestEnemy

For the defensive tactic, a fighter uses the following rules:

- **if** $(RoundNumber \leq 1)$ **and** $(PotionFR = yes)$

  then DrinkPotionFR
- **if** $(HP < 50)$ **and** $(PotionHealing = yes)$

  then DrinkPotionH
- **if** $(HP \geq 50)$ **then** AttackClosestEnemy

Finally, a wizard playing the defensive tactic uses the rules:

- **if** $(HP < 50)$ **and** $(PotionHealing = yes)$

  then DrinkPotionH
- **if** $(HP \geq 50)$ **and** $(Level2SpellDF = yes)$

  then CastLevel2DFSpellCenterEnemy
- **if** $(HP \geq 50)$ **and** $(Level3SpellDF = yes)$

  then CastLevel3DFSpell
- **if** $(HP \geq 50)$ **and** $(Level1SpellDF = yes)$

  then CastLevel1DFSpellClosestEnemy
- **if** $(HP \geq 50)$ **then** AttackClosestEnemy

Since the tactics are written as rules, we extracted from these rules a set of corresponding attributes and actions. Using these attributes together with a couple more attributes added, we managed to create all possible examples that are covered by these rules, forming training examples suitable for processing by a decision tree as VFDT. It is important to notice that the extra attributes added are in fact irrelevant for the tactics we want to learn, but including then in the data generated is important because in a real CRPG a big number of irrelevant attributes to a given tactic will be forming the training examples, and the ML algorithm needs to deal with these attributes. Table 1 presents all the attributes extracted from the rules and their types. Table 2 presents the extra attributes created.

Table 1: *Set of attributes extracted from the rules forming the tactics.*

Attribute	Type	Explanation
HP	Numeric (From 1 to 100)	Health percentage of the character
RoundNumber	Numeric (From 1 to 20)	Indicates the actual round number
PotionHealing	Binary	Indicates if the character has a healing potion
PotionFR	Binary	Indicates if the character has a fire resistance potion
Level1 Spell DG	Binary	Indicates if the character has a damaging spell of level 1
Level2 Spell DG	Binary	Indicates if the character has a damaging spell of level 2
Level3 Spell DG	Binary	Indicates if the character has a damaging spell of level 3
Level1 Spell DF	Binary	Indicates if the character has a defensive spell of level 1
Level2 Spell DF	Binary	Indicates if the character has a defensive spell of level 2
Level3 Spell DF	Binary	Indicates if the character has a defensive spell of level 3
TypeClosestEnemy	Categorical	Indicates the type of the closest enemy
Action	Categorical	The chosen action (target attribute)

Not all of these attributes are used for generating the dataset corresponding to a given opponent and tactic. The set of attributes used to generate each combination of opponent/tactic are:

- Fighter Offensive: HP, PotionHealing, PotionFR, PotionFA, SelfInfluenceStatus, LocationStatus, DistClosestEnemy, Action.
- Wizard Offensive: HP, PotionHealing, PotionFR, PotionFA, SelfInfluenceStatus, LocationStatus, DistClosestEnemy, TypeClosestEnemy, Level1 Spell DG, Level2 Spell DG, Level3 Spell DG, Action.
- Fighter Defensive: HP, RoundNumber, PotionHealing, PotionFR, PotionFA, SelfInfluenceStatus, LocationStatus, DistClosestEnemy, Action.
- Wizard Defensive: HP, RoundNumber, PotionHealing, PotionFR, PotionFA, SelfInfluenceStatus, LocationStatus, DistClosestEnemy, TypeClosestEnemy, Level1 Spell DF, Level2 Spell DF, Level3 Spell DF, Action.

To meet the second objective, we constructed a dataset containing examples of the offensive and defensive tactic for a fighter opponent. The dataset simulates a data stream where the target concept changes over time from one tactic to another. The goal of the algorithms investigated is to detect the change in tactics, represented in the examples arriving in the stream, and respond accordingly by means of adapting the actual decision model. We investigated two methods for detecting and reacting to concept drift, namely, the DDM method and its extension EDDM. The ensemble methods presented in Section 5 were also experimented with this simulated stream because, as explained before, ensembles are known to be quickly adaptable to changes in the distribution of the examples.

In the experiments conducted we show that the algorithms for data stream mining investigated in this work can in fact adapt the decision model to the changes in tactics in the stream of examples.

7 Empirical Evaluation

In this section we empirically evaluate VFDTs in the simulated context of a CRPG. In order to reach the objectives of this work, we constructed four datasets representing the two static tactics for each type of opponent (fighter and wizard). Section 7.1 presents the experiments conducted to meet the first objective and analyse VFDT in a stationary situation of learning one combat tactic. Section 7.2 presents the experiments where the second objective is met, and show how VFDTs can be used in non-static environments where the combat tactics change over time.

7.1 Learning Static Tactics

For the first experiment, we want to evaluate how fast, by means of number of training examples, the VFDT-NB algorithm can effectively learn the tactic represented in the examples, by means of the accuracy of the induced model. It is important for the context of this work to evaluate the behavior of VFDT-NB when a small number of training examples are available to the algorithm, because in a real game situation, one cannot afford expending a very long time to learn a predictive model. It is known that VFDT-NB is not severely affected by the order in which the examples are presented to the tree. However, this is true when we have a reasonable number of training examples. Because in the initial state the algorithm may be affected by the order of the examples, to have a more confident value for the accuracy of the classifier, we carried out the following procedure.

First, we generated all the possible examples for one type of opponent's tactic, for example a fighter playing the offensive tactic. After that, we randomly selected 1000 examples for training and 500 examples for testing. We generated 10 pairs of training/test sets. The VFDT-NB algorithm was run in each one of this training sets and tested in the corresponding test set in order to calculate a mean of the accuracy.

The metodology for evaluating VFDT-NB in a pair of training/test sets was the same one used by Gama et al [Gama et al. 2006], evaluating the accuracy on the test set with different numbers of training examples observed by the algorithm.This methodology allows to create a learning curve that takes into account the increasing number of examples available to the algorithm. This type of analysis is important for a data stream algorithm in order to prove its anytime property, i.e, the capacity of the algorithm of having a decision model ready to make predictions at any time.

The VFDT-NB algorithm used in this experiment is the implementation from the MOA package [MOA - University of Waikato]. The parameters used were $\delta = 5\times10^{-3}$, $\tau = 0$, $n_{min} = 1$ and the minimum number of examples observed before permiting Naive-Bayes application also equal to 1.

Tables 3 and 4 present the results obtained.

As can be seen by Table 3, for the fighter opponent the VFDT-NB has a high predictive accuracy with a relative small number of examples. This was expected because the fighter's offensive tactic is a simple tactic, depending only on two attributes, namely HP and PotionHealing. For the wizard's offensive tactic, the tree takes longer to achieve the same accuracy because this tactic is more complicated. In spite of this, it is possible to have a model that predicts the wizard's tactic with approximately 85% of accuracy with 100 examples, which, in fact, is not a very large number of examples. It is interesting to notice that as the number of training examples increases, the mean accuracy increases and the standard deviation decreases accordingly, showing that as the number of examples increases, the order in which these examples are presented to the tree has decreasing impact on the accuracy.

When the number of training examples is too small, for example from 5 to 100, the accuracy of the predictions rely basically on the Naive-Bayes classification realized on the leaves, which in this case is the root of the tree. Because of the nature of the VFDT-NB algorithm, the tree will not decide for installing a split node if there is not enough statistical support in favor of that split. As we are using a high value for δ the tree will take longer to decide for

Table 2: *Set of extra attributes created.*

Attribute	Type	Explanation
DistClosestEnemy	Numeric (From 1 to 20)	Distance to the closest enemy
SelfInfluenceStatus	Binary	Indicates if the character is under the influence of a spell
LocationStatus	Binary	Indicates if the character is within the area covered by a cloud effect
PotionFA	Binary	Indicates if the character has a free action potion

Table 3: *Mean accuracy of VFDT-NB, standard deviation, maximum and minimum values of accuracy observed for the offensive tactic. Values calculated for different numbers of training examples and test set containing* 500 *examples.*

#Ex	Fighter				Wizard			
	Accuracy (%)	**SD**	**Max**	**Min**	**Accuracy (%)**	**SD**	**Max**	**Min**
5	76.58	1.75	79.40	74.00	30.14	10.80	43.60	12.20
10	80.06	4.33	90.00	75.40	38.54	10.65	53.40	19.00
15	83.90	7.25	95.20	75.40	46.56	6.20	53.20	35.40
20	86.76	6.02	95.80	75.40	55.42	8.36	66.00	39.80
25	88.74	4.75	96.60	82.40	60.48	5.89	74.20	54.20
30	89.04	4.03	95.00	80.80	65.84	5.18	74.20	59.60
40	91.28	3.88	97.60	84.40	72.40	3.10	77.00	68.00
50	92.50	2.29	95.40	88.60	76.44	4.74	82.40	70.00
100	96.20	1.34	98.80	93.80	84.80	3.08	87.80	80.20
200	96.16	1.27	98.20	94.60	89.78	3.59	94.20	83.40
300	96.04	1.20	97.60	94.00	92.18	2.04	94.80	88.80
400	96.74	1.57	99.00	94.00	93.58	1.47	96.00	91.60
500	96.80	1.47	99.00	94.00	93.90	1.59	96.20	92.00
1000	97.51	2.04	100.00	94.60	95.64	1.01	96.60	94.00

a split and therefore, will have to observe a reasonable amount of examples.

The results in Table 4 are related to the defensive tactic. For the fighter, the tree has a predictive accuracy of approximately 90% with 50 examples observed. The tree takes slightly more time to learn this tactic than it needed to learn the offensive tactic for the fighter opponent, because for this type of character the defensive tactic is a little more complicated than the offensive tactic. The wizard's defensive tactic is learned faster by the VFDT-NB than the wizard's offensive tactic because the defensive tactic is a simplification of the offensive one. One interesting point can be observed from the wizard's results in Table 4. The accuracy value with 300 examples observed by the tree decreases significantly in comparison to the accuracy with 200 examples and then starts to recover and increases as the number of examples continue to increase. The explanation for this fact is that the tree decided to install a decision node somewhere between 200 and 300 examples. Since this is a new node, the number of examples that reached this node is relatively small and, consequently, the Naive-Bayes classification at this node will not be as effective as it was in the father of this node simply because it had more examples to consider when making a prediction. As the number of examples that reach the new node increases, the classification tends to be more accurate, as can be observed in the results.

7.2 Learning with Changing Tactics

For the second experiment, we used the datasets generated for the fighter playing the offensive tactic and the fighter playing the defensive tactic. We created a new dataset using the examples contained in this two datasets in the following way. The first sequence of examples in the dataset were all of the offensive tactic (1000 examples), then, we simulated a concept drift by adding a new sequence of examples from the defensive tactic (1500 examples). After that sequence of examples, a new sequence of 1500 examples from the offensive tactic was added to the dataset, and so on, until reaching 5.500 examples. Two test sets, each containing 50 examples, were used to test the model at intervals of 500 training examples. Each test set contains examples of one of the tactics and they are used interchangeably to test the model. The test set containing examples from the defensive tactic is used to test the model right before the examples from the defensive tactic arrive at the stream, so the tree did not see any defensive tactic examples before. Then we keep this test set until reaching the moment where the offensive tactic examples arrive at the stream. At this moment, we switch for the offensive test set. The process repeats until the end of the examples.

We investigated the performance of VFDT-NB without drift detection, VFDT-NB with DDM and VFDT-NB with EDDM. We also investigated four ensemble methods, all of them using VFDT-NB as the base classifier: the Bagging method by Oza and Russel with 5 trees (OzaBag 5 VFDT-NB), the OzaBag ADWIN method with 5 trees (OzaBag ADWIN 5 VFDT-NB), the Bagging with Adaptive-Size Hoeffding Trees using 5 ASHT (OzaBag ASHT) and the Adaptive Hoeffding Option Tree (ASHOT). Although it is more common to use dozens of classifiers in the ensembles, we opted for using only a small number of classifiers. This was motivated by to reasons. The first is that, in the data stream scenario, smaller ensembles are more appropriate. Ensembles for data streams are often parallelized to work online. The higher the number of base classifiers in the ensemble, the higher the time expended in communication between processing units. The second reason is related to applying these methods in a real game. In a real computer game, one cannot afford the AI consuming a lot of processing power. It is straightforward that with more classifiers in the ensemble, more processing power will be needed. Therefore, the final purpose was to check whether we could achieve good results with relative small computational processing.

Figure 1 presents the results obtained by VFDT-NB, VFDT-NB with EDDM and VFDT-NB with EDDM.

Using the Simpson's Rule for calculating the area under the curves we concluded that the method with better performance is VFDT-NB with EDDM. Figure 2 presents the results of the ensemble methods. Between these methods and VFDT-NB with EDDM, again VFDT-NB with EDDM is the one with better performance by means of area under the curve. The methods OzaBag 5 VFDT-NB, OzaBag ADWIN and ASHOT achieved better accuracy then the VFDT-NB method with no drift detection. Methods VFDT-NB with DDM and OzaBag ASHT had inferior performance then all the other methods investigated.

Table 5 presents the calculated areas under the curves for each classification method.

Analysing the curve of VFDT-NB with EDDM in Figure 2, we can observe how this method is rearranging its decision model when the concept underlying the training examples changes. For exam-

Table 4: *Mean accuracy of VFDT-NB, standard deviation, maximum and minimum values of accuracy observed for the defensive tactic. Values calculated for different numbers of training examples and test set containing* 500 *examples.*

#Ex	Fighter				Wizard			
	Accuracy (%)	**SD**	**Max**	**Min**	**Accuracy (%)**	**SD**	**Max**	**Min**
5	71.90	2.02	75.00	68.80	32.58	6.85	38.60	21.20
10	72.80	1.80	75.80	70.40	40.82	9.72	49.80	23.00
15	75.72	5.23	88.40	71.00	52.46	7.22	66.40	40.00
20	79.00	6.60	93.00	71.20	60.40	7.10	69.60	49.00
25	80.38	5.47	92.20	73.20	65.58	5.55	74.40	57.20
30	84.12	3.66	90.60	78.80	70.22	5.29	75.20	60.20
40	88.66	3.55	93.80	82.40	77.36	5.34	86.00	69.60
50	89.98	4.46	95.00	83.20	84.08	5.25	93.20	77.00
100	92.86	3.48	97.00	87.00	91.60	4.37	96.60	83.00
200	94.86	1.88	97.00	91.20	95.82	1.04	97.20	93.80
300	95.64	1.97	98.80	93.00	77.12	20.73	97.60	34.60
400	96.10	1.43	98.40	93.60	84.44	12.21	94.20	58.60
500	96.44	1.54	98.60	94.20	92.78	3.87	96.20	85.00
1000	96.68	1.25	98.40	94.60	94.52	2.90	97.20	87.40

Table 5: *Calculated areas under the curves for each method investigated.*

Method	**Area**
VFDT-NB	455
VFDT-NB with DDM	448
VFDT-NB with EDDM	460
OzaBag 5 VFDT-NB	459
OzaBag ADWIN 5 VFDT-NB	459
OzaBag 5 ASHT	447
ASHOT	458

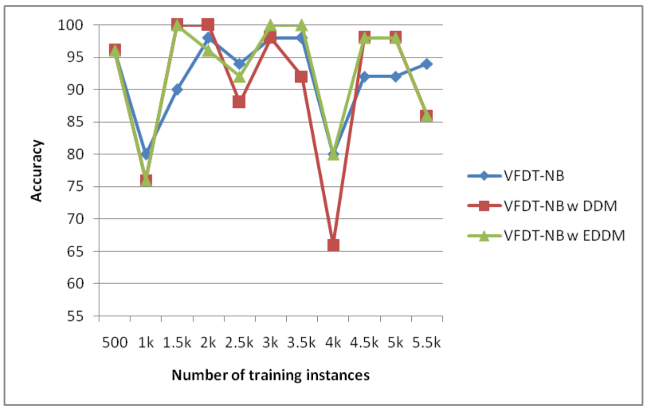

Figure 1: *Accuracy (%) of the classification methods in a stream of data changing between offensive and defensive tactic for a fighter opponent.*

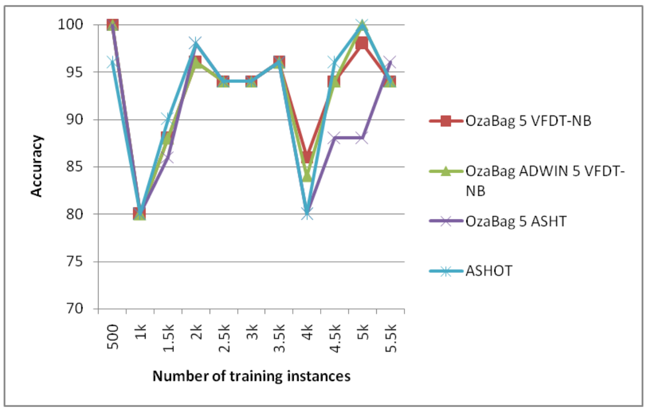

Figure 2: *Accuracy (%) of the ensemble methods in a stream of data changing between offensive and defensive tactic for a fighter opponent.*

ple, after seeing 1000 examples, the concept in the stream changes. This is reflected in the graph with a decrease in classification accuracy. The next examples in the stream represent this new context and, therefore, the tree needs to be rearranged. At 1500 examples observed the accuracy of the decision model has already recovered, representing that the change in concept was detected and that the algorithm reacted accordingly adjusting the decision model.

These results are interesting because they show that methods designed to work in non-stationary problems really perform better when the concept defining the stream of examples can change over time, as is the case in these simulated changes of tactics in a CRPG.

8 Conclusion

In this work, we investigated the use of algorithms for mining data streams in the context of CRPGs. More specifically, we analysed the VFDT-NB algorithm and a collection of ensemble methods using VFDTs in simulated CRPG data. Since these algorithms are incremental and designed to work online, we believe that they represent a natural approach for learning combat tactics in CRPGs. The approach used in this work is based on Player Modelling, where the primary goal is to improve the opponent player by allowing it to adapt to the human player strategies. Our experimental results showed that VFDT-NB accurately learned static combat tactics using a small number of training examples. Building a decision model in few learning trials is essential to any learning algorithm to be used in a real game environment, because this applications cannot afford expending time and resources to learn an accurate model. In our simulated data, VFDT-NB, due to its incremental nature, presented a promising performance for computer games learning. Besides, this algorithm does not need to store the examples in memory.

The experiments also showed that VFDTs designed to react to concept drift can, in fact, adapt their learning models to a context where the player changes tactics from time to time. This feature can be usefull for online learning, where traditional ML algorithms that learn static models are not suitable. According to the experimental results, the algorithms investigated can detect a change in the combat tactic being used. Furthermore, they can adapt the decision model to these new data in order to better represent the target concept.

These results show the potential of applying algorithms for data stream mining in real games. Further studies need to be conducted in a real game scenario, for example, using a CRPG simulator and not only simulated data, as we did in this work. Additionally, a deeper investigation in how these algorithms perform in an online situation can bring powerful insights in how to adapt the methods to allow their use in a real game. Another possible future research direction is to initialize the VFDT algorithm with a previously constructed tree.

References

BAENA-GARCÍA, M., DEL CAMPO-ÁVILA, J., FIDALGO, R., BIFET, A., GAVALDÀ, R., AND MORALES-BUENO, R. 2006. Early Drift Detection Method. In *Fourth International Workshop on Knowledge Discovery from Data Streams*.

BAKKES, S., SPRONCK, P., AND VAN DEN HERIK, J. 2009. Rapid and Reliable Adaptation of Video Game AI. *IEEE Transactions on Computational Intelligence and AI in Games 1*, 2, 93–104.

BIFET, A., AND GAVALDÀ, R. 2007. Learning from Time-changing Data with Adaptive Windowing. In *SIAM International Conference on Data Mining*, 443–448.

BIFET, A., HOLMES, G., KIRKBY, R., AND GAVALDÀ, R. 2009. New Ensemble Methods for Evolving Data Streams. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, 139–148.

CATLETT, J. 1991. *Megainduction: Machine Learning on Very Large Datasets*. PhD thesis, Basser Department of Computer Science, University of Sydney, Sydney, Australia.

CROCOMO, M. K., MIAZAKI, M., AND SIMÕES, E. V. 2007. Algoritmos Evolutivos para a produção de NPCs com Comportamentos Adaptativos. In *Anais do Simpósio Brasileiro de Jogos Para Computador e Entretenimento Digital*, 44–53.

DOMINGOS, P., AND HULTEN, G. 2000. Mining High-Speed Data Streams. In *Proceedings of the sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, 71–80.

FÜRNKRANZ, J., 2007. Recent Advances in Machine Learning and Game Playing.

GAMA, J., AND RODRIGUES, P. P. 2007. Data Stream Processing. In *Learning from Data Streams: Processing Techniques in Sensor Networks*. Springer, 25–38.

GAMA, J., AND RODRIGUES, P. P. 2009. An Overview on Mining Data Streams. In *Studies in Computational Intelligence*. Springer Berlin/Heidelberg, 29–45.

GAMA, J., MEDAS, P., CASTILLO, G., AND RODRIGUES, P. P. 2004. Learning with Drift Detection. In *Proceedings of the 17th Brazilian Symposium on Artificial Intelligence*, Springer, 286–295.

GAMA, J., FERNANDES, R., AND ROCHA, R. 2006. Decision Trees for Mining Data Streams. *Intelligent Data Analysis 10*, 23–45.

KOHAVI, R., AND KUNZ, C. 1997. Option Decision Trees with Majority Votes. In *Proceeding of the International Conference on Machine Learning*, 161–169.

MILLINGTON, I. 2006. *Artificial Intelligence for Games*. Morgan Kaufmann.

MOA - UNIVERSITY OF WAIKATO. MOA Data Stream Mining. (Acess on 17/06/2010).

OZA, N., AND RUSSEL, S. 2001. Online Bagging and Boosting. *Artificial Intelligence and Statistics*, 105–112.

PFAHRINGER, B., HOLMES, G., AND KIRKBY, R. 2007. New options for hoeffding trees. In *Proceedings of the 20th Australian joint conference on Advances in artificial intelligence*, Springer-Verlag, 90–99.

SPRONCK, P., SPRINKHUIZEN-KUYPER, I., AND POSTMA, E. 2003. Online Adaptation of Game Opponent AI in Simulation and in Practice. In *Proceedings of the 4th International Conference on Intelligent Games and Simulation (GAME-ON 2003)*, 93–100.

SPRONCK, P. 2005. *Adaptive Game AI*. PhD thesis, Masstricht University.

SZITA, I., PONSEN, M., AND SPRONCK, P. 2009. Effective and Diverse Adaptive Game AI. *IEEE Transactions on Computational Intelligence and AI in Games 1*, 16–27.