Redução de Tráfego de Jogos Distribuídos Através da Predição de Movimento Baseada em Aprendizado de Máquina

Pâmela de Assis Beltrani, Aurora T. R. Pozo, Elias P. Duarte Jr.

[1]Departamento de Informática – Universidade Federal do Paraná (UFPR)
Curitiba – PR – Brasil

{pabeltrani,aurora,elias}@inf.ufpr.br

Abstract. *Distributed game players maintain a consistent vision of the positions of each other by periodically exchanging messages. Besides the fact that these messages can suffer delays that cause rendering inconsistencies, they also represent an overhead on the network. In order to avoid sending these messages, strategies for movement predicton, such as Dead Reckoning, can be employed. In this work, we present a movement prediction strategy based on machine learning. The strategy consists of two phases. In the first phase, a learning model classifies whether Dead Reckoning is able to predict the new position correctly or not. In case of a mistake, another learning model is used to predict the new direction. The proposed strategy has been applied to the World of Warcraft game. To evaluate the new strategy, the learning models were implemented with the Weka tool using real trace data. Results show that a traffic reduction of 66.41% on average, which is superior to other approaches.*

Resumo. *Em jogos distribuídos* multiplayer *os jogadores mantém uma visão consistente das posições uns dos outros através da troca periódica de mensagens. As mensagens de atualização, além de poderem sofrer atrasos de entrega que causam inconsistências na renderização, representam uma sobrecarga na rede. Estratégias de predição de movimento, como o tradicional algoritmo* Dead Reckoning*, evitam a troca de mensagens quando acertam. Neste trabalho, apresentamos uma nova estratégia para a predição de movimento baseada em aprendizado de máquina. A estratégia consiste de duas fases. Na primeira, um modelo de aprendizado classifica se o* Dead Reckoning *acerta ou erra. Em caso de erro é utilizado um outro modelo de aprendizado para prever a nova posição. A estratégia proposta foi aplicada para o jogo* World of Warcraft*. Para a avaliação, os modelos de aprendizado foram implementados na ferramenta Weka. Foram utilizados dados de* traces *reais do jogo. Resultados mostram que a redução de tráfego foi de 66,41% em média, superior a outras abordagens.*

1. Introdução

Os jogos eletrônicos estão atingindo níveis de popularidade extremamente elevados. A *Entretaiment Software Association* (ESA), aponta que em 2015 são 155 milhões os norte-americanos que jogam video-games freqüentemente [1]. A ESA também aponta que de cada cinco casas norte-americanas em quatro delas existe algum dispositivo eletrônico que é

[1]http://www.theesa.com/facts/pdfs/esa_ef_2014.pdf

utilizado para jogar. Essa presença cada vez mais marcante dos jogos na vida das pessoas se reflete diretamente na comercialização dos mesmos. Por exemplo, o jogo *Grand Theft Auto V* (GTA V), lançado em setembro de 2013, vendeu 11,21 milhões de cópias no dia de seu lançamento [2]. Outro caso de sucesso é o jogo *Call of Duty: Black Ops II* (CODII) que vendeu um total de 13,62 milhões de cópias [3]. Diversos destes jogos são distribuídos e *multiplayer*, permitindo que vários jogadores interajam através da Internet.

Em jogos distribuídos, os jogadores devem ter uma visão consistente do estado do mundo do jogo, de forma que múltiplos jogadores têm a mesma visualização. Para que essa visão seja consistente, as informações sobre o estado do jogo são transmitidas através da troca periódica de mensagens. Dentre as informações que precisam ser trocadas, estão as informações sobre a posição de cada avatar, a entidade que representa um jogador dentro do jogo, e a sua movimentação. Entretanto, estas mensagens de atualização de movimento dos avatares, além de poderem sofrer atrasos que causam inconsistências ou ainda saltos abruptos de renderização, representam uma sobrecarga na rede.

Tradicionalmente, o algoritmo *Dead Reckoning* é utilizado para prever a movimentação que os avatares realizam e, quando acerta, evita a troca de mensagens [Standards Committee on Interactive Simulation 1995]. Por exemplo, considere que um cliente deve prever a posição de um avatar de um outro jogador; em ambos os clientes o *Dead Reckoning* é executado e somente ocorre uma troca de mensagens quando é constatado que a previsão está incorreta. Para fazer essa previsão, o *Dead Reckoning* utiliza as leis da física e assume que esta movimentação será em linha reta. Em outras palavras, o *Dead Reckoning* não considera que o avatar poderá se movimentar em uma nova direção. Como em jogos é desejavel que o jogador tenha muita interação, e é comum que essa interação seja através da movimentação seu avatar, é notória a baixa taxa de precisão do *Dead Reckoning* nestes ambientes. Uma abordagem baseada em computação bio-inspirada para melhorar a taxa de precisão do *Dead Reckoning* é o algoritmo *AntReckoning* [Yahyavi et al. 2012]. O problema desta abordagem é que necessita que um especialista no jogo configure os multiplos parâmetros utilizados pelo *AntReckoning* o que dificulta a utilização desta abordagem na prática.

Neste trabalho, apresentamos uma nova estratégia para a predição da movimentação de avatares em jogos distribuídos baseada em aprendizado de máquina. Esta estratégia tem duas fases bem definidas. Na primeira fase, um modelo de aprendizado classifica se o *Dead Reckoning* acerta ou erra a predição de um determinado avatar. Em caso de acerto, o *Dead Reckoning* é utilizado pra fazer a previsão. Porém, em caso de erro, a segunda fase é executada. Nessa segunda fase, é utilizado um novo modelo de aprendizado para fazer a previsão da nova direção de movimento. Os modelos de aprendizado são construídos utilizando os algoritmos: *Local Weighted Learning* [Atkeson et al. 1997], *Bootstrap Agregating* (Bagging) [Ripley 1996], *Multilayer Perceptron* [Quinlan 1987] e *Reduced Error Pruning Tree* [Opitz and Maclin 1999].

A estratégia proposta foi aplicada para o jogo *World of Warcraft*[4] . Os modelos de aprendizado foram construídos utilizando uma nova base de dados de traces do jogo.

[2]http://www.guinnessworldrecords.com/news/2013/10/confirmed-grand-theft-auto-breaks-six-sales-world-records-51900/

[3]http://www.vgchartz.com/game/70716/call-of-duty-black-ops-ii/

[4]World of Warcraft: http://us.battle.net/wow/pt/

Esta nova base de dados foi construída utilizando dados de *traces* que estão disponíveis publicamente [Shen et al. 2014]. Esta base contém informações sobre o campo de visão de cada um dos jogadores e apresenta informações sobre quais jogadores e NPCs (*Non-Playable Characters*) estão visíveis por determinado jogador em um instante de tempo.

Resultados mostram que a estratégia proposta obtém uma taxa de acerto médio de 76,60% para a primeira fase; e de 51,02% para a segunda fase. Destaca-se o algoritmo *Bagging*, que obtém uma taxa de acerto para a primeira e a segunda fases de 81,10% e 73,37%, respectivamente. Os testes executados mostram também uma taxa de acerto final médio de 66,41%, observa-se que nesta mesma base o algoritmo *Dead Reckoning* obteve a taxa de acerto de 57,89%. Destaca-se novamente o algoritmo *Bagging* obteve uma taxa de acerto final de 78,76%. Estes resultados mostram que a utilização de aprendizado de máquina para a predição da movimentação de avatares jogos distribuídos é uma alternativa muito atraente. Em comparação com o AntReckoning, a sua a taxa de sucesso é semelhante, cerca de 30% de melhoria sobre o *Dead Reckoning*. Entretanto, a grande vantagem da estratégia proposta é que não necessita de um especialista para setar os múltiplos parâmetros utilizados por aquele modelo. Utilizando aprendizado de máquina, os parâmetros são "automaticamente aprendidos".

O restante deste trabalho está organizado da seguinte forma. Na Seção 2 são apresentados conceitos básicos de jogos distribuídos. Em seguida, na Seção 3, os algoritmos *Dead Reckoning* e o *AntReckoning* são apresentados. Na Seção 4 é apresentada uma breve introdução aos conceitos de aprendizado de máquina. Na Seção 5 a estratégia proposta é descrita. Os resultados obtidos para o jogo *World of Warcraft* são apresentados na Seção 6. Por fim, a conclusão segue na Seção 7.

2. Jogos: Definições Básicas

Nessa seção são apresentados, de forma sucinta, os principais os conceitos de jogos que são necessários para a compreensão da contribuição deste trabalho. Nos jogos baseados em modelo de avatar o jogador interage com o jogo por meio de um personagem representativo, o avatar, que existe em uma localização do mundo do jogo [Claypool and Claypool 2010]. Pelo avatar o jogador consegue apenas ver e interagir com o que está presente em suas imediações. Nos jogos com avatar pode existir uma diferença de perspectiva, esta pode ser tanto em primeira quanto em terceira pessoa. Existem alguns jogos que permitem inclusive a escolha da visão utilizada.

Uma classificação comum em jogos considera com quantos jogadores uma pessoa pode interagir simultaneamente agrupando assim, os jogos em *singleplayer*, *multiplayer* e *massively multiplayer online*. Nos jogos *singleplayer* um jogador não interage com outros jogadores. Na segunda categoria estão os jogos *multiplayer*, que permitem uma experiência com vários jogadores, podendo ser em forma de disputa ou cooperação, e normalmente proporciona uma forma de comunicação social que está quase sempre ausente em jogos *singleplayer*. Os jogos *multiplayer* podem ser jogados tanto em rede quanto localmente.

Por último, também existem os *massively multiplayer online games* (MMOG) que são capazes de suportar grandes quantidades de jogadores simultaneamente em uma mesma partida. É importante ressaltar que essa classificação considera o número de jogadores interagindo ao mesmo tempo, portanto, mesmo que existam 7,5 milhões de pessoas

jogando simultaneamente o jogo *League of Legends*[5] (LoL) por existir um limite máximo de 10 jogadores em um partida, LoL é considerado um jogo *multiplayer* e não um MMOG. Neste trabalho, são chamados de jogos distribuídos todos os jogos *multiplayer* em rede e os *massively multiplayer online*.

Em jogos *multiplayer*, os clientes mantêm cópias dos objetos presentes no mundo do jogo. Quando um jogador interage com algum desses objetos, todos os outros clientes que são afetados por essa ação devem ter o seu estado interno do jogo atualizado. Uma solução encontrada para este problema é a utilização de áreas de interesse (*Area of Interest* - AOI). O jogador só recebe as informações que são relevantes sob seu ponto de vista, normalmente considerando a posição e o campo de visão do avatar dentro do mundo do jogo. Dessa forma, os jogadores apenas mantêm réplicas dos objetos que são interessantes para eles. Neste trabalho consideramos que a área de interesse é modelada como uma esfera que envolve o avatar do jogador. Desta forma, o jogador pode receber informações inclusive sobre objetos que não são visíveis.

Apesar da grande quantidade de gêneros e estilos dos jogos atuais, além dos avatares, existem apenas três tipos de entidades com as quais o jogador pode interagir: os objetos imutáveis, os objetos mutáveis e os personagens não jogáveis (NPCs - *Non Playable Character*) [Yahyavi and Kemme 2013]. Os objetos imutáveis são objetos criados *offline* e que nunca são alterados durante o jogo. Esses objetos normalmente são inicializados quando o jogo começa, por exemplo, objetos de cenário, o mapa do jogo, os modelos dos inimigos, dentre outros. Os objetos mutáveis são instâncias de objetos com os quais o jogador pode interagir e usar em si mesmo ou em outros jogadores. Exemplos desses objetos são poções que o jogador pode utilizar para recuperar a saúde de seu avatar. Os NPCs (personagens não jogáveis) são personagens ou ainda avatares que não são controlados por um jogador, são normalmente controlados por uma inteligência artificial. São muito utilizados para contar uma história, entregar missões ao jogador ou simplesmente trazer mais realismo para o ambiente fazendo ações rotineiras.

3. Técnicas de Predição de Movimento em Jogos

Um dos grandes problemas nos jogos *multiplayer* é garantir a consistência da visão dos multiplos jogadores. Por exemplo, considere dois avatares A e B. O avatar A está parado, enquanto o avatar B executa um movimento retilíneo uniforme de direção perpendicular à direção de A. Após alguns instantes, o avatar B muda sua direção de movimentação. O jogador B precisa enviar uma mensagem ao jogador A para informar sua nova posição.

Em jogos *Massively Multiplayer Online*, alguns com milhões de jogadores simultaneos, a quantia de mensagens transmitidas pode ser muito significativa. Técnicas de predição de movimento podem ser usadas para evitar o envio destas mensagens. Um exemplo de técnica preditiva é o *Dead Reckoning*.

O *Dead Reckoning* estima a nova posição tendo em vista a posição anterior e equações de movimento [Standards Committee on Interactive Simulation 1995]. Ambos os jogadores executam o algoritmo e uma mensagem só é enviada se o erro for superior ao um valor aceitável. As variáveis mais importantes para o *Dead Reckoning* são: a última posição conhecida x_t da entidade, a velocidade $v_t = \dot{x}_t$, a aceleração $a_t = \ddot{x}_t$ e t

[5]League of Legends: http://br.leagueoflegends.com/

que representa o instante de tempo. Outras informações que ajudam a estimar as forças que atuam nessa entidade também podem ser incluídas. Para objetos representados em um espaço tridimensional também são considerados a orientação, a velocidade angular e possivelmente a aceleração angular. A partir da definição dessas variáveis a trajetória de um objeto pode ser descrita utilizando as equações de movimento, mais especificamente pela da segunda lei de Newton, que utiliza a aceleração a_t de um objeto, a massa m e as forças F_i que esse objeto está submetido. Podendo assim fazer uma estimativa da posição de um determinado jogador.

Utilizando essas estimativas, cada jogador deve manter além de sua posição a posição estimada pelo *Dead Reckoning*. Quando a posição real e a posição estimada estiverem com uma diferença maior que o limiar aceitável, o jogador deve comunicar aos outros jogadores sua posição atual real. Cada jogador também deve manter um modelo de posição/orientação para todas as outras entidades que estiverem dentro de sua área de interesse. Essas posições/orientações estimadas das outras entidades são utilizadas para renderizar a posição/orientação das entidades no jogo. Porém, em jogos é muito comum que as regras físicas sejam relaxadas e mudanças drásticas de movimento são permitidas. É importante ressaltar que os clientes só recebem informações sobre os objetos que estão dentro de sua AOI em MMOGs.

Em [Yahyavi et al. 2012] é apresentado o algoritmo preditivo *AntReckoning* que é inspirado em colônia de formigas e tem como objetivo a utilização de modelos de interesse para prever a movimentação dos jogadores. As principais contribuições do *AntReckoning* são a incorporação do interesse do jogador na equação de movimento utilizada pelo *Dead Reckoning* e prover um arcabouço que utiliza feromônios para a modelagem considerando aspectos temporais e espaciais do interesse dos jogadores. Para incorporar o interesse dos jogadores na estimativa da posição desses avatares foi criado um modelo de forças de atração e repulsão. A intensidade das forças exercidas pelos pontos de atração (*Point of Interest* - POI) dependem de sua atratividade que pode ser determinada ou ainda aprendida. Esses pontos de interesse variam de um jogo para outro, mas um exemplo são os itens no jogo que atraem os jogadores, especialmente se os itens forem valiosos ou caso o jogador precise deles urgentemente. Da mesma forma, locais que não são seguros ao jogador, ou ainda locais em que existam inimigos muito fortes comparados ao nível do avatar.

Esses POIs são tratados pelo *AntReckoning* como formigas que geram feromônios para modelar a atração dos jogadores. Ao longo do tempo esses feromônios se propagam pelo mundo do jogo e a concentração acaba decaindo. Assim, naturalmente, fatores temporais e a geometria do mundo do jogo são simulados.

Assim como é comum em muitos jogos, o *AntReckoning* assume que o mundo do jogo é divido em células que não se sobrepõem. Denota-se por C o tamanho da célula na unidade do mundo do jogo. A gerência dos feromônios e as forças de atração exercidas por eles dependem da granularidade da célula: para cada avatar P, o cliente Q executa o algoritmo de *Dead Reckoning*, Q calcula a concentração de feromônio em cada célula e a respectiva soma das forças exercidas. Por questão de escalabilidade, apenas as células de uma região limitada em volta de P, chamada de zona de atração e denotada por R, é considerada no cálculo. É importante ressaltar que como os feromônios se propagam, mesmo POIs que não estão dentro de R são considerados. Outra questão é o fato de que

mesmo no *AntReckoning* o jogador também precisa executar o algoritmo para si mesmo, já que precisa saber quando deve se comunicar com os outros jogadores sobre seu posicionamento, considerando o limiar η escolhido.

4. Aprendizado de Máquina

Aprendizado de máquina é um ramo da inteligência artificial que faz intersecção entre a computação e a estatística, que busca compreender quais são as leis fundamentais que regem os processos de aprendizado com o objetivo de construir sistemas que aprendam para melhorar seus resultados de forma automática [Mitchell 2006].

Os algoritmos de aprendizado de máquina podem realizar tarefas tais como: Classificação, Predição, Clusterização e Associação [Camilo 2009]. As tarefas de Classificação e Predição são vistas como tarefas supervisionadas, enquanto as tarefas de Clusterização e Associação são tarefas não supervisionadas. As tarefas de Classificação têm como objetivo catalogar valores de variáveis qualitativas. Quando se busca estimar um valor numérico de uma variável a tarefa pode ser chamada de Regressão. As tarefas de Predição têm como objetivo prever um valor possível para uma variável no futuro. As predições numéricas tem como objetivo final a previsão para variáveis contínuas; no caso de variáveis discretas, devem ser utilizadas técnicas de classificação.

As tarefas de Agrupamento funcionam como descrito a seguir. Dado um conjunto de dados de entradas, são construídos agrupamentos, também chamados de *clusters*, que contém os registros mais semelhantes. Portanto, um dado é considerado similar aos elementos pertencentes ao mesmo *cluster* e, normalmente, para o cálculo da similaridade entre as entradas são utilizadas medidas de distâncias tradicionais. Por último, a Associação consiste em identificar o relacionamento dos ítens mais freqüentes em um determinado conjunto de dados.

4.1. Matriz de Confusão

A partir da obtenção dos resultados apresentados por um algoritmo de aprendizado de máquina, é necessário fazer um teste para verificar o quão confiável é o modelo. A matriz de confusão, apresentada em [Provost et al. 1997], contém informações sobre como as entradas foram classificadas, determinando se o valor previsto corresponde ao valor correto. Para obter uma matriz de confusão, é apresentado um conjunto de dados com suas classificações e para cada dado apresentado se obtém a classificação predita e compara-se com a classificação real. Todos os dados são contabilizados e os totais são apresentados em uma matriz. A matriz de confusão de um classificador binário é apresentado na tabela 1. Nessa matriz temos as colunas apresentando em quais categorias os valores foram classificados pelo algoritmo, enquanto nas linhas temos as categorias que essas instâncias realmente pertencem. É possível observar que existem duas categorias em que as instâncias podem estar e existem quatro grupos em que uma instância pode ser representada nessa matriz. As instâncias que pertencem aos grupos *a* e *d* foram corretamente classificadas. Já as instâncias pertencentes aos grupos *b* e c estão sendo classificadas de forma errônea.

As instâncias pertencentes aos grupos de *a*, *b*, *c* e *d* são chamadas respectivamente de Verdadeiras Negativas (True Negatives - TN); Falsas Positivas (False Positives - FP); Falsas Negativas (False Negatives - FN); e Verdadeiras Positivas (True Positives - TP).

	Classificado: Negativo	Classificado: Positivo
Realidade: Negativo	TN = a	FP = b
Realidade: Positivo	FN = c	TP = d

Tabela 1. Matriz de confusão exemplo.

5. A Abordagem Proposta

Nessa Seção é apresentada a proposta para a predição de movimento baseada em aprendizado de máquina. Inicialmente é descrito o jogo *World of Warcraft*. Em seguida, os *traces* reais utilizados são descritos bem como a nova base que foi construída baseada estes *traces*. Por fim, a nova estratégia é apresentada.

5.1. O Jogo *World of Warcraft* e a Cidade *IronForge*

Para realizar esse trabalho foi escolhido o jogo *World of Warcraft*[6] (WoW). Este jogo pertence ao gênero de MMOG e constitui uma grande parcela do mercado deste gênero com 100 milhões de contas criadas, sendo jogado em 224 países e territórios, segundo a *Blizzard Entertainment* [7]. Esse gênero tem algumas características intrínsecas: a necessidade de uma maneira de medir quantitativamente o progresso do jogador, a interação social com outros jogadores, a possibilidade de personalização dos avatares e uma arquitetura do sistema construída focando a presença de um grande volume de jogadores. Em WoW, os jogadores podem observar o seu progresso no jogo por meio do avanço do nível do avatar, nos pontos gastos em habilidades, em equipamentos adquiridos, na riqueza obtida dentro do jogo ou ainda pelas de conquistas obtidas fazendo certas tarefas. Para progredir nesse jogo, os avatares batalham com monstros ou ainda fazem missões que são dadas por NPCs. Ambas ações podem ser feitas em grupo ou de forma solitária e normalmente provêm recompensas como pontos de experiência que são utilizados para que os avatares subam de nível.

Em particular a predição de movimento é feita utilizando dados colhidos na cidade de *Ironforge* devido a disponibilidade de traces do jogo nesta cidade. A cidade de *Ironforge* é considerada uma cidade capital da Aliança no jogo *World of Warcraft*. Os jogadores de *World of Warcraft* podem participar de uma das duas facções presentes, a Aliança ou a Horda. Dependendo do servidor em que esses jogadores estejam jogando, é possível fazer batalhas do tipo jogador-versus-jogador entre jogadores de facções rivais. É importante ressaltar que como a cidade analisada é uma das capitais da Aliança, é improvável a presença de jogadores com avatares da Horda, e portanto geralmente o comportamento dos jogadores não assume formas agressivas. O que garante isso é que inclusive os NPCs da cidade irão atacar avatares da Horda caso sejam vistos, se tornando assim um ambiente extremamente agressivo aos avatares da Horda. Outra informação importante sobre *Ironforge* é que ela é a cidade inicial para todos os jogadores da raça anã, portanto, é uma cidade que é visitada tanto por avatares poderosos quanto por avatares fracos. Em *World of Warcraft*, a partir do nível 20, os jogadores podem utilizar montarias com uma velocidade maior do que o avatar caminha e acima do nível 60 os jogadores podem inclusive utilizar montarias que voam.

[6]http://us.battle.net/wow/pt/

[7]Blizzard Entertainment: http://us.battle.net/wow/pt/blog/12346804/world-of-warcraft-azeroth-by-the-numbers-1-28-2014

Os NPCs desta cidade podem desempenhar uma das seguintes funções: *Auctioneer, Banker, Battlemaster, Collector, Innkeeper, Merchant, Quartermaster, Quest Giver, Stable Master, Trainer, Weapon Master*. O interesse por qualquer um desses NPCs é muito individual de um jogador, pois um jogador que tenha a profissão de cozinheiro não se interessará pelo mercante que vende ítens para forja ou forneça treinamento em alfaiataria, por exemplo. Além disso, um jogador com um nível muito alto, não se interessará por missões para iniciantes.

5.2. *Traces* Reais do Jogo Utilizadas

Neste trabalho foram utilizados traces reais do jogo WoW disponíveis em duas bases publicas na Internet: *WoWPosition* e *WoWHead*. A partir dos dados destas bases, foi construída uma nova base utilizada no desenvolvimento da proposta apresentada no presente trabalho.

A base de dados *WoWPosition* [Shen et al. 2014] é uma das bases que foi utilizadas neste trabalho. Essa base de dados apresenta informações sobre a posição dos jogadores. Entretanto, apresenta unicamente esta informação, não tendo qualquer outro tipo de informação sobre nível, raça ou ainda classe de um determinado avatar. Destaca-se que essa base de dados não contém qualquer informação que também possa ser utilizada para identificar o avatar. Dentre os mapas que foram analisados para a construção desta base foi escolhida os dados do arquivo "Ironforge-1d". Essa base contém informações de 1302 jogadores, os quais foram coletadas na cidade de *Ironforge*.

A base de dados *WoWPosition* contém mais de 30.000 entradas e é composta por uma tabela com os seguintes campos: *timestamp, id, x, y, z e ângulo de visão*, descritos a seguir. O campo *timestamp* é a quantidade de segundos que se passaram desde do primeiro segundo do primeiro dia do ano de 1970. O campo *id* apresenta um número de identificação utilizado internamente para identificar um jogador. Os jogadores foram anonimizados e, portanto, não é possível extrair mais informações, além das fornecidas na própria base sobre os mesmos. Os campos *x, y e z* são os dados da posição do jogador dentro do jogo e o *ângulo de visão* representa para qual direção o jogador está olhando.

Outra base de dados que foi utilizada neste trabalho é a base *WowHead* [8] . A partir dessa base de dados é possível extrair informações sobre os NPCs presentes no mapa da cidade de *Ironforge*. Essa base de dados é composta por uma tabela com 4 campos: *Nome, ThottbotXPosition, ThottbotYPosition e FunçãoNPC*. O primeiro campo é composto pelo nome do NPC, o segundo e o terceiro campos são responsáveis pelo armazenamento da posição do NPC no mapa dentro do sistema de coordenadas do denominado *Thottbot* e o último campo descreve a função deste NPC. Foram criadas as seguintes categorias para descrever a função de um NPC dentro do jogo: *Auctioneer, Banker, Battlemaster, Collector, Innkeeper, Merchant, Quartermaster, Quest Giver, Stable Master, Trainer, Weapon Master*.

O jogo *World of Warcraft* é um jogo em 3 dimensões, entretanto a base *WoWHead* faz uma conversão da posição tridimensional de todos o NPCs para um mapa com 2 dimensões [9]. Por conta dessa característica a coordenada que representa a altura é ignorada.

[8] WoWHead: http://www.wowhead.com

[9] Coordenadas Thottbot: http://www.wowwiki.com/Mapcoordinates

O sistema de coordenadas utilizado consiste de uma tupla (x, y) com os valores variando de 0 até 100 e representam uma localização dentro deste mapa com duas dimensões. Nesse sistema, o canto superior esquerdo representa a posição $(0, 0)$ e o canto inferior direito representa a posição $(100, 100)$, dessa maneira, o centro do mapa é representado pela posição $(50, 50)$.

5.3. Construção da Nova Base

Utilizando as base de dados *WoWPosition* e *WoWHead* foi construída uma nova base que contém os seguintes campos: *ThottbotX*, *ThottbotY*, o *AnguloVisual*, *AnguloMovimentacaoAnterior*, os dados dos outros jogadores e NPCs presentes no campo de visão, o *AnguloMovimentacao* e *DeadReckoning*. Os campos *ThottbotX* e *ThottbotY* indicam a posição do jogador no sistema de coordenadas do *Thottbot*, enquanto o campo *AnguloVisual* é responsável por indicar para qual direção que o jogador que está sendo análisado está observando. Os campos com as informações dos outros jogadores e dos NPCs e serão descritos no próximo parágrafo. Os dois últimos campos são o *AnguloMovimentacao* e o *DeadReckoning*. O campo *AnguloMovimentacao* descreve qual o ângulo em que o jogador se movimentou no respectivo *timestamp* e o campo *DeadReckoning* indica se o algoritmo *Dead Reckoning* acerta aquele caso ou não. Em outras palavras, o campo *DeadReckoning* deve ser utilizado para o treinamento dos modelos de aprendizagem na primeira fase da nova estratégia. Enquanto o campo *ÂnguloMovimentacao* é utilizado durante a segunda fase da estratégia proposta. Portanto, quando se está na etapa de o aprendizado para determinar se o *Dead Reckoning* irá acertar ou não, durante a primeira fase, o campo *AnguloMovimentacao* é ignorado. Já quando está sendo feito o aprendizado do modelo da segunda fase, quando de fato é determinado o novo ângulo de movimentação do jogador, o campo *DeadReckoning* deverá ser ignorado.

Os campos dos jogadores e NPCs são: *X*, *Y*, *Distância*, *Angulo* e *Função*. Os campos *X* e *Y* apresentam a posição destes jogadores ou NPCs no sistema de coordenadas do *Thottbot*. O campo *Distância* mostra a distância entre o jogador analisado e um outro jogador ou NPC, enquanto o campo *Angulo* indica o ângulo entre os dois. O campo *Função* é exclusivo dos NPCs e mostra qual a função de um determinado NPC no jogo. A quantidade de campos presentes na base de dados precisa ser fixa, assim no máximo N jogadores e NPCs são tratados dentro do campo de visão de cada jogador. Caso existam mais que N jogadores ou NPCs no campo de visão, apenas os N mais próximos são tratados.

Para determinar se um avatar ou NPC está dentro do campo de visão de um jogador é necessário verificar o ângulo entre eles e qual é o ângulo que o jogador está observando. Os dados fornecidos pela base *WoWPosition* estão no sistema de posicionamento real utilizado pelo jogo, entretanto todos os NPCs têm o posicionamento apresentado pela base *WoWHead*. Portanto, para poder ser feita a análise do campo de visão de um jogador, todos os avatares tiveram suas coordenadas de posição convertidas para o sistema da base *WoWHead*. Destaca-se que esta decisão foi tomada considerando que na base *WoWHead* não existe a coordenada de altura. Outra informação relevante é que não temos qualquer informação sobre o posicionamento da câmera de cada jogador ou ainda sua direção de observação. O que temos é a informação sobre qual o ângulo de determinado avatar está rotacionado. Assim verificamos se um NPC ou avatar está dentro do campo de visão utilizando dois testes. Primeiramente, é testado se o avatar/NPC está dentro da AOI do

jogador. Em [Shen et al. 2014] é apresentado que a área de interesse dos jogadores é representada por um círculo de raio de 100 unidades do jogo, essa medida está sendo utilizada para o valor de d. Em seguida, é testado se o avatar/NPC foi utilizado o valor α para determinar o ângulo de abertura do campo de visão do jogador. Neste trabalho foi utilizado o valor de 80º para α por ser próximo ao valor do ângulo de abertura do campo de visão de seres humanos.

Foi realizada uma "limpeza da base", utiliando os modelos de aprendizado construídos visando eliminar campos ruidosos e aqueles que não podem ser interpretados individualmente. Portanto, a base de dados ficou com os seguintes campos: *AnguloVisual*, *AnguloMovimentaçãoAnterior*, *AnguloMovimentacao* e *DeadReckoning* e dados dos jogadores e dos NPCS são compostos por *Distancia*, *Angulo* e *Funcao*.

5.4. Estratégia Proposta

A estratégia proposta deve ser utilizada sempre que for necessário renderizar um determinado avatar de um jogador por outro. A estratégia proposta consiste em duas fases bem definidas. Na primeira fase, algoritmos de aprendizado de máquina são utilizados para gerar um modelo que classifica se o *Dead Reckoning* acerta ou erra na predição de movimento. O *Dead Reckoning*, considera que um jogador se movimenta em linha reta e utiliza regras físicas para prever a nova posição. Caso contrário, isto é, quando a primeira fase indica mudança de direção é executada a segunda fase. Na segunda fase, algoritmos de aprendizado de máquina são utilizados para gerar um modelo que faz a predição do ângulo de movimento do avatar.

Para implementar a estratégia foi utilizada a ferramenta Weka [10] (Waikato Environment for Knowledge Analysis) em sua versão 3.6.12. Foram escolhidos experimentalmente quatro algoritmos, apresentados a seguir, para fazer a predição de movimento: *Reduced Error Pruning Tree* (REPTree), *Local Weighted Learning* (LWL), Bootstrap Aggregating (*Bagging*) e uma rede *Multilayer Perceptron*.

O algoritmo *Reduced Error Pruning Tree* (REPTRee) é baseado em árvores de decisão [Quinlan 1987]. O algoritmo *Local Weighted Learning* (LWL) faz um sistema de votação ponderada [Atkeson et al. 1997]. O *Bootstrap Aggregating* (*Bagging*) usa a estratégia de gerar várias versões de um mesmo classificador utilizando diferentes versões do conjunto de treinamento original [Opitz and Maclin 1999]. Por fim, foi utilizada uma rede neural do tipo *Multilayer Perceptron*. [Ripley 1996].

6. Resultados Experimentais

Inicialmente foram criadas duas bases de dados com os dados de um período de tempo de 1 hora cada: A primeira base trata da primeira hora e a segunda base tratou da oitava hora. Foi utilizado o valor N igual a 7. As informações dessas bases foram sintetizadas na tabela 2. Essa tabela é composta por sete campos. O primeiro campo identifica a base e o segundo campo indica o total de entradas presente nesta base. O segundo e o terceiro campo indicam respectivamente o total de acertos e erros do algoritmo *Dead Reckoning* nesta base. O quarto campo indica quantos avatares distintos estão sendo retratados nesta base de dados. Os últimos dois campos indicam o *Timestamp* inicial e final que foram utilizados para a construção destas bases de dados.

[10] http://www.cs.waikato.ac.nz/ml/weka/

Identificação da base	Total de Entradas	Total de acertos	Total de erros	Avatares distintos	Timestamp Inicial	Timestamp Final
1	19870	61,17%	38,83%	164	63445137060	63445140659
2	4885	44,54%	2709 (55,46%	43	63445165860	63445169459
Somatório	24755	57,89%	10425(42,11%	200	-	-

Tabela 2. Informações sobre as bases de dados geradas.

Através da tabela 2 é possível retirar algumas informações sobre o comportamento do *Dead Reckoning*. Nessas bases a taxa de acerto médio do algoritmo *Dead Reckoning* é de 57,89%, e , portanto, a taxa de erro é de 42,11%.É importante ressaltar que cada uma das entradas presentes na base de dados equivale ao registro da posição de um avatar dentro do jogo *World of Warcraft*. Como apresentado anteriormente, as situações em que o *Dead Reckoning* erra sua previsão, são as situações em que devem haver troca de mensagens comunicando a nova direção. Portanto, nessas bases de dados em 42.11% entre um segundo e outro houve ao menos uma troca de mensagens para comunicar a mudança da movimentação de jogadores.

Além do comportamento do algoritmo *Dead Reckoning*, é possível também observar que os jogadores têm um comportamento diferente dependendo do horário. Observa-se que na tabela 2 é significativa a diferença entre a quantidade de avatares distintos entre a primeira e a oitava hora apesar de ter sido utilizada a mesma quantidade de timestamps. Na primeira hora, que foi utilizada para construir a base de dados 1, existem 164 avatares distintos. Enquanto na oitava hora, que foi utilizada para a construção da base 2, existem apenas 43 avatares distintos. Como havia menos jogadores é esperado que existam menos dados de movimentação na base 2 do que na 1.

São apresentados resultados de três experimentos. O primeiro experimento têm por objetivo determinar quando *Dead Reckoning* erra ou acerta. O segundo experimento tinha como objetivo determinar, nos casos em que o *Dead Reckoning* erra, qual será o novo ângulo de movimentação. No terceiro experimento, a segunda fase foi executada apenas utilizando os dados que foram classificados como Verdadeiros Positivos e Falsos Negativos na primeira fase.

6.1. Primeiro Experimento

O primeiro experimento tinha como objetivo verificar se é possível utilizar algoritmos de aprendizado de máquina para classificar quando o algoritmo *Dead Reckoning* erra a previsão. Os resultados do primeiro experimento são apresentados na tabela 3. É possível observar que, em média os algoritmos obtiveram uma taxa de acerto de 79,29%. Além disso, é possível observar que o algoritmo *Bagging* obteve o melhor resultado ao obter uma taxa de acerto de 83,17%. Ressalta-se que os algoritmos, em média, classificam 26,59% das entradas em que o *Dead Reckoning* acertaria de forma errada, ou seja, 26,59% das entradas são Falsos Positivos. Dessa forma, 26,59% dos casos em que o *Dead Reckoning* acertaria na predição são enviados para a segunda fase da estratégia proposta de forma incorreta. Também observa-se que em 20,70% das vezes em que o *Dead Reckoning* erra a predição, os algoritmos de aprendizado apontam que *Dead Reckoning* irá acertar. Assim, para todas essas entradas a predição será errada tanto utilizando o *Dead Reckoning* quanto na nova estratégia. A conclusão obtida neste experimento é que com a média da taxa de acerto entre os algoritmos sendo 76.60% verificou-se que é viável utilizar aprendizado de máquina para a primeira fase do modelo.

Algoritmo	Treinamento	Teste	TP	FP	TN	FN	Taxa de acerto
MultilayerPerceptron	1	2	81,35%	22,98%	77,02%	18,65%	79,42%
Bagging com REPTree	1	2	83,17%	21,43%	78,57%	16,83%	81,10%
LWL	1	2	69,92%	39,82%	61,18%	30,08%	65,85%
REPTree	1	2	82,74%	23,16%	76,84%	17,26%	80,04%
Média	-	-	79,29%	26,59%	73,40%	20,70%	76,60%

Tabela 3. Resultados interpretados da aprendizagem do campo DeadReckoning.

6.2. Segundo Experimento

O segundo experimento tinha como objetivo determinar se é possível utilizar algoritmos de aprendizado de máquina para prever qual será o novo ângulo de movimentação de um jogador, quando o jogador muda de direção. Para isso foram removidas todas as entradas em que o *Dead Reckoning* faz a previsão correta. Neste experimento foi considerado que uma saída está correta se houver um erro menor que 15º, e portanto menor que 5%. Os resultados obtidos pelos algoritmos para a predição do novo ângulo de movimento podem ser observados na tabela 4. Nessa tabela também é possível observar que os algoritmos erraram 43,48% de todas as entradas que deveriam ir para segunda fase. Em outras palavras, os algoritmos de aprendizado erraram na predição de qual será o novo ângulo de movimentação em 43,48% dos casos. Como os algoritmos estão sendo utilizados para fazer uma predição, é possível observar que a média do erro absoluto foi de 29,62%. Novamente, como foi no teste anterior, o algoritmo de *Bagging* obteve o melhor desempenho, tendo um erro médio absoluto de 15,35% e ele conseguiu prever corretamente para 83,78% das entradas o novo ângulo de movimento. Estes resultados demonstram a viabilidade do aprendizado do novo ângulo xdos jogadores no jogo *World of Warcraft*.

Algoritmo	Treinamento	Teste	Erro Absoluto	Coef. de Cor.	Saídas Corretas	Saídas Erradas
MultilayerPerceptron	1	2	33,44 %	0,81	1067(49,04%)	1109(50,96%)
Bagging com REPTree	1	2	15,35 %	0,92	1823(83,78%)	353(16,22%)
LWL	1	2	52,50 %	0,79	230 (10,56%)	1946(89,44%)
REPTRee	1	2	17,20 %	0,91	1799(82,67%)	377(17,33%)
Média	-	-	29,62 %	0,85	56,50%	43,48%

Tabela 4. Resultados para aprendizagem do campo *AnguloMovimento*.

6.3. Terceiro Experimento

O terceiro experimento foi elaborado considerando que se o classificador errar durante a primeira fase, mas a previsão na segunda fase estiver correta, a saída estará correta. Assim foram construídas bases de dados utilizando todas as entradas em que cada um dos algoritmos classificou como entradas que deveriam ir para a segunda fase. Ou seja, para cada um dos algoritmos foi construída uma base utilizando as entradas Falsas Negativas quanto Verdadeiras Positivas indicadas pelos próprios modelos da primeira fase.

Observa-se que nesta tabela que os algoritmos erram em média em 48,97% de todas as entradas que foram para segunda fase. No teste anterior, foi observado que eles erravam em 43,48% dos casos. Portanto, houve uma variação de 5,49% na quantidade de saídas erroneamente classificadas. Observa-se também que o erro absoluto médio obtido durante o terceiro teste é de 34,88%, enquanto no segundo teste foi de 29,62%. Assim a variação foi de 5,26% entre o segundo e o terceiro teste.

Algoritmo	Total de Entradas	Erro Absoluto	Coef. de Cor.	Saídas Corretas	Saídas Erradas
MultilayerPerceptron	2167	36,75 %	0,76	1021 (47,12%)	1146 (52,88%)
Bagging com REPTree	2193	21,86 %	0,87	1609 (73,37%)	584 (26,63%)
LWL	2272	55,79 %	0,67	312 (13,73%)	1960 (86,26%)
REPTRee	2237	25,12 %	0,85	1563 (69,87%)	674 (30,13%)
Média	2717.25	34,88 %	0,78	51,02%	48,97%

Tabela 5. Resultados obtidos no terceiro teste.

Algoritmo	1a Fase		2a Fase	
	Caso 1	Caso 2	Caso 3	Caso 4
MultilayerPerceptron	2211 (45,26%)	507 (10,37%)	1021 (20,90%)	1146 (23,45%)
Bagging com REPTree	2239 (45,83%)	453 (9,27%)	1609 (32,93%)	584 (11,95%)
LWL	1827 (37,40%)	786 (16,09%)	312 (6,38%)	1960 (40,12%)
REPTRee	2191 (44,85%)	457 (9,35%)	1563 (31,95%)	674 (13,79%)
Média	43,33%	11,27%	23,04%	22,32%

Tabela 6. Resultados obtidos para as entradas presentes na base 2.

Na tabela 6 são apresentados os resultados de como uma instância pode ser classificada utilizando esta estratégia. O casos contabilizados como Caso 1 são casos que o modelo indica que o *Dead Reckoning* irá acertar e de fato ele acerta. Já os casos classificados como Caso 2 são casos que o modelo indica que o *Dead Reckoning* irá acertar, porém ele erra. Durante a segunda fase, o modelo pode prever corretamente a nova direção do jogador ou não, estes casos estão sendo computados respectivamente pelo Caso 3 e Caso 4 na tabela. Desta forma, é possível observar na tabela 6, que o algoritmo *Bagging* com o algoritmo REPTree obteve o melhor resultado se comparado a todos os outros algoritmos. De todas as entradas presentes na base 2, o algoritmo *Bagging* previu corretamente, na primeira fase, que 45.83% das entradas o algoritmo *Dead Reckoning* seria suficiente. Além disso, de todas as entradas na base 2, o algoritmo *Bagging* previu corretamente o novo ângulo de movimentação para 32,93% de todas as entradas fornecidas. Desta maneira, somando a porcentagem de casos 1 e 3 temos uma taxa de acerto de 78,76% de todas as entradas presentes na base 2. Destaca-se que nesta base, como foi apresentado na tabela 2, o algoritmo *Dead Reckoning* obteve uma taxa de acerto de 44,54%.

7. Conclusão

Neste trabalho foi apresentada uma proposta de predição da movimentação de avatares em jogos distribuídos *multiplayer* baseada em aprendizado de máquina. Para a validação da estratégia proposta foi construída uma base de dados, baseada em traces do jogo *World of Warcraft*, a partir das bases disponíveis publicamente *WoWPosition* e *WoWHead*. A nova base de dados construída contém informações sobre jogadores e NPCs que estão presentes no campo de visão de um determinado jogador. Além destas informações, também estão disponíveis dados sobre o posicionamento deste jogador.

Observou-se, que a estratégia proposta obtém uma taxa de acerto médio de 76,60% durante a primeira fase. Na segunda fase obtém uma taxa de acerto de 51,02%. O algoritmo de aprendizagem de máquina que obtém o melhor desempenho é o Bagging. Esse algoritmo obtém uma taxa de acerto durante a primeira e a segunda fase respectivamente de 81,10% e 73,37%. Estes resultados mostram que a utilização de aprendizado de máquina para a predição da movimentação de avatares jogos distribuídos é uma alternativa muito atraente Em comparação com uma outra abordagem relacionada da literatura, o AntReckoning, a taxa de sucesso é semelhante, cerca de 30% de melhoria sobre o *Dead*

Reckoning. Entretanto, a grande vantagem da estratégia proposta é que não necessita de um especialista para configurar os múltiplos parâmetros utilizados por aquele modelo. Utilizando aprendizado de máquina, os parâmetros são "automaticamente aprendidos".

Entre os trabalhos futuros um item importante é construir uma base de dados ainda com mais informações que a construída, melhorando a predição. Além disso, os modelos propostos devem ser aplicados para outros jogos distribuídos.

Referências

Atkeson, C. G., Moore, A. W., and Schaal, S. (1997). Locally weighted learning for control. In *Lazy learning*, pages 75–113. Springer.

Camilo, C. O. (2009). Mineração de dados: Conceitos, tarefas, métodos e ferramentas.

Claypool, M. and Claypool, K. (2010). Latency Can Kill: Precision and Deadline in Online games. In *Proceeding MMSys '10 Proceedings of the first annual ACM SIGMM conference on Multimedia systems*.

Mitchell, T. M. (2006). *The discipline of machine learning*, volume 17. Carnegie Mellon University, School of Computer Science, Machine Learning Department.

Opitz, D. and Maclin, R. (1999). Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research*, pages 169–198.

Provost, F., Fawcett, T., and Kohavi, R. (1997). The case against accuracy estimation for comparing induction algorithms. In *In Proceedings of the Fifteenth International Conference on Machine Learning*, pages 445–453. Morgan Kaufmann.

Quinlan, J. R. (1987). Simplifying decision trees. *International journal of man-machine studies*, 27(3):221–234.

Ripley, B. D. (1996). *Pattern recognition and neural networks*. Cambridge university press.

Shen, S., Brouwers, N., Iosup, A., and Epema, D. (2014). Characterization of human mobility in networked virtual environments. In *Proceedings of Network and Operating System Support on Digital Audio and Video Workshop*, page 13. ACM.

Standards Committee on Interactive Simulation (1995). IEEE Standard for Distributed Interactive Simulation - Application Protocols . In *Proceedings of IEEE Standard 1278.1-1995*, New York, NY, USA.

Yahyavi, A., Huguenin, K., and Kemme, B. (2012). Interest modeling in games: the case of dead reckoning. *Multimedia Systems*, pages 255–270.

Yahyavi, A. and Kemme, B. (2013). Peer-to-Peer Architectures for Massively Multiplayer Online Games: A Survey. *ACM Computing Surveys*, 46 Issue 1:Article 9.