Detecção de programas de computador que automatizam as ações do jogador em jogos do gênero MMORPG

Anderson Chaves Faria, Leonardo Nascimento

11053613, 11051613

1. Introdução

Os programas de computador que automatizam as ações do jogador (bots) têm sido um dos maiores problemas em jogos do gênero *Massively Multiplayer Online Role Playing Game* (MMORPG), uma vez que seu uso contra jogadores humanos é desleal e faz com que as pessoas abandonem o jogo [Mishima 2013]. Deste modo, desenvolver modos de identificar bots é de extrema importância para manter a base de jogadores estável.

Neste trabalho será utilizada a linguagem Python, a biblioteca de aprendizado de máquina scikit-learn[1] e uma base de dados artificial, gerada a partir de características identificadas em bots do jogo Tibia, para construir um classificador capaz de detectar o uso de bots no jogo.

2. Base de dados

A base de dados artificial possui mil amostras, sendo que vinte por cento do total são bots. Os atributos foram escolhidos a partir de características identificadas em bots do jogo Tibia, sendo o principal deles é o Magebot, que permite que o trapaceador escolha quais serão as características do bot, por exemplo, relacionadas à coleta de itens, autocura, e combate.

Sabendo que o objetivo do jogo é aumentar o nível do personagem a partir da coleta de itens liberados quando monstros são mortos e não morrer durante o processo, a hipótese é a de que bots têm ações mais mecânicas do que jogadores humanos e, apesar de existirem técnicas para incluir ações erráticas aos bots, o comportamento de jogadores humanos é ímpar.

A partir da coleta de dados em um jogo real, e sabendo quais dos jogadores eram bots, foi possível identificar que a média dos intervalos de ação de um bot costuma ser inferior à de um jogador humano para reagir a um evento, e também que a mediana da diferença entre esses intervalos é menor se o gatilho não for aleatório.

Assim, foram criadas distribuições normais ou uniformes de tal modo que o comportamento de cada atributo pudesse ser emulado. Os seguintes atributos foram considerados para produzir a base de dados artificial, e são relativos a um intervalo de cinco minutos.

- **itens coletados** (*collected_items*): metade dos bots coletam mais itens do que jogadores no intervalo, principalmente porque eles são capazes de matar monstros com maior eficiência ou explorar o mapa de forma eficiente.
- **tempo médio para coletar item** (*avg_time_to_collect_item*): o tempo médio em

[1] https://scikit-learn.org/stable/index.html

milissegundos entre um item aparecer no cenário e ser coletado é muito menor para bots do que os demais jogadores, pois estes não dependem do mouse para executar a ação. Vale considerar que alguns bots configuram um tempo de atraso para evitar banimento.

- **diferença de tempos para coletar item** (*delta_time_to_collect_item*): os bots são consistentes em suas ações e a diferença entre elas é pequena, ao passo que os jogadores não possuem padrão definido e a diferença costuma ser grande.
- **limiar de cura** (*heal_threshold*): sempre que vida do personagem chega a um nível predefinido os bots começam a cura, enquanto jogadores esperam o que pensam ser o melhor momento.
- **tempo médio para iniciar cura** (*avg_time_to_start_healing*): o tempo médio em milissegundos entre o último dano sofrido e o início da cura costuma ser baixo para bots, a menos que exista um atraso configurado, enquanto jogadores observam o estado do personagem antes de iniciar a cura.
- **diferença de tempos para iniciar cura** (*delta_time_to_start_healing*): os jogadores avaliam cada situação antes de iniciar uma cura, enquanto bots executam o que foi configurado.
- **inimigos mortos** (*killed_enemies*): os bots costumam matar um número alto de inimigos ou serem modestos para evitar banimento, enquanto jogadores podem ter desmpenho muito melhor ou muito pior do que os bots.
- **tempo com fome** (*hungry_time*): o tempo absoluto em segundos do período que o personagem ficou com fome pode mostrar que uma rotina automática está sendo executada, pois normalmente os jogadores esquecem de executar essa ação e o personagem fica com fome por longos períodos.

3. Metodologia

A metodologia utilizada segue o procedimento sugerido pela documentação do scikit-learn, descrito na Figura 1.

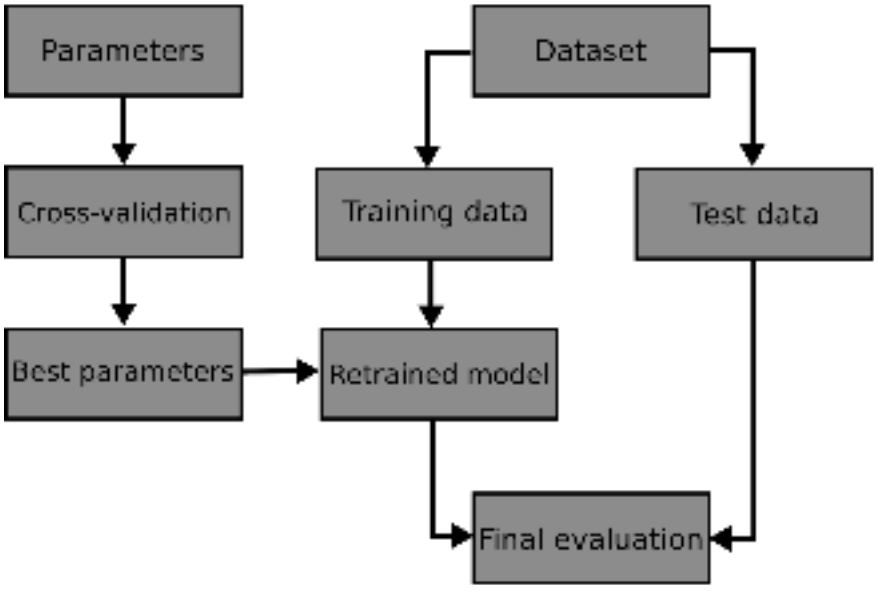

Figura 1. Diagrama da metodologia utilizada para escolher o classificador.

Definiu-se que o classificador que apresentasse melhor desempenho dentre os escolhidos seria avaliado. Assim, foi feita a validação cruzada com 10-folds e os melhores parâmetros com base no score padrão foram encontrados.

Aplicando-os ao classificador, foi possível obter seus scores de acurácia, precisão, f-measure, recall e AUC. Entretanto, não é possível declarar nenhum deles como superior para a base de dados construída, haja vista que os scores estão bastante próximos uns dos outros e, em boa parte dos casos, dentro da margem de erro.

Deste modo, escolhemos os classificadores SVC com função de kernel polinomial e árvore de decisão para serem analisados, o primeiro por ter a maior AUC e o menor desvio padrão, e o segundo por ter a maior acurácia e o menor desvio padrão, como mostra a Tabela 1, mesmo que isso não tenha significância estatística.

Tabela 1. Desempeho dos classificadores avaliados.

Classificador	**Acurácia**	**Precisão**	**F-Measure**	**Recall**	**AUC**
DecisionTree	0.994±0.008	0.981±0.031	0.985±0.020	0.990±0.030	0.997±0.004
SVC poly	0.992± 0.010	0.981± 0.032	0.980± 0.024	0.980± 0.024	0.999± 0.002
SVC rbf	0.992± 0.010	0.990± 0.020	0.979± 0.025	0.970± 0.040	0.998± 0.005
Linear SVC	0.991± 0.011	0.985± 0.032	0.977± 0.028	0.970± 0.033	0.999± 0.003
SVC linear	0.990± 0.010	0.985± 0.023	0.974± 0.026	0.965± 0.039	0.999± 0.003
K-Neighbors	0.990± 0.010	0.985± 0.023	0.974± 0.026	0.965± 0.039	0.999± 0.003

4. Classificadores escolhidos

O SVC com kernel polinomial é capaz de caracterizar atributos como polinômios a partir dos originais, permitindo o aprendizado de modelos não lineares, o que não parece ser um caso, uma vez que o SVC com kernel linear também conseguiu obter um desempenho semelhante.

Quando avaliamos o classificador SVC(C=3.0, degree=2, gamma='scale', kernel='poly') após o treino em uma base de dados completamente não vista, o desempenho é satisfatório, pois os scores não são muito diferentes do que foi encontrado no benchmark.

Deste modo, os vetores de suporte definidos durante o treino são robustos e capazes de generalizar bem o problema de classificação.

A árvore de decisão também obteve um excelente desempenho, e seu diagrama, representado na Figura 2, é capaz de informar qual a relevância dos atributos presentes na base dados, e verificar que os atributos ‘inimigos mortos’ e ‘tempo com fome’, não são utilizados.

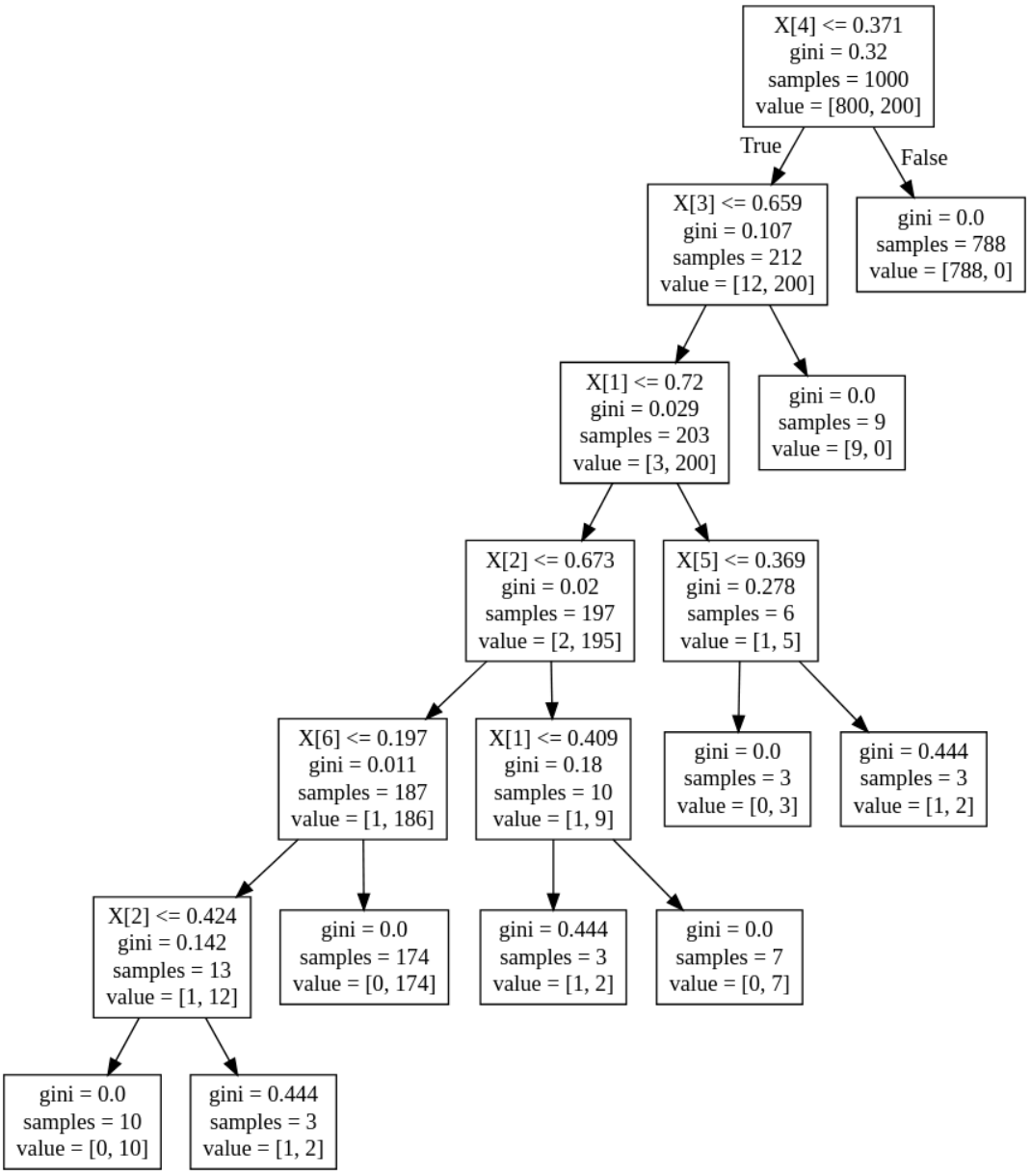

Figura 2. Diagrama do classificador árvore de decisão com melhor desempenho para a base de dados antibot.

5. Considerações Finais

Foi bastante difícil escolher quais atributos seriam utilizados e qual tipo de dado melhor representaria os atributos na base de dados. Por vezes considerou-se utilizar atributos nominais para representar as medidas de consistência, por exemplo.

Definir um período de coleta que fizesse sentido para banir um bot e sem prejudicar jogadores também gerou bastante discussão e ainda não se sabe se o período de cinco minutos representa o ideal.

Ainda, começar a trabalhar com a base de dados é essencial para entender problemas que podem ter ocorrido durante a criação da base.

Referências

Mishima, Y., Fukuda, K., & Esaki, H. (2013). An analysis of players and bots behaviors in MMORPG. In 2013 IEEE 27th International Conference on Advanced Information Networking and Applications (AINA) (pp. 870-876). IEEE.

Hilaire, S., Kim, H. C., & Kim, C. K. (2010, June). How to deal with bot scum in MMORPGs?. In 2010 IEEE International Workshop Technical Committee on Communications Quality and Reliability (CQR 2010) (pp. 1-6). IEEE.

Chen, K. T., Jiang, J. W., Huang, P., Chu, H. H., Lei, C. L., & Chen, W. C. (2009). Identifying MMORPG bots: A traffic analysis approach. EURASIP Journal on Advances in Signal Processing, 2009, 3.