Uma Arquitetura de Aprendizagem para a Condução Automática de Veículos

André Pinz Borges, Richardson Ribeiro, Allan Rodrigo Leite, Osmar Betazzi Dordal, Bruno Giacomet, Bráulio Coelho Ávila, Fabrício Enembreck, Edson Emílio Scalabrin

Programa de Pós-Graduação em Informática Aplicada (PPGIa),
Pontifícia Universidade Católica do Paraná, Brasil
{andre.borges, richard, allan, osmarbd, bruno.giacomet, avila, fabricio, scalabrin}@ppgia.pucpr.br

Abstract. In this paper we propose a new learning architecture for autonomous vehicle conduction using intelligent agent technologies. The system's agents have a set of resources to generate the action policy; vehicle and road characteristics and a knowledge base that contains the rules of conduction. The agent perception is guaranteed by a set of sensors, which provides information like speed, position and brake conditions. The agent main behavior is to execute actions involving: increase, maintain or decrease the speed. The main focus of this research is to induce rules of conduction from historical data travel. Such rules build a classifier that is used to select the actions with the intention to generate the conduction plain. Experimental results show that the developed architecture using the classifier proposed increase the efficiency of the autonomous vehicle conduction.

1 Introdução e Motivação

O uso de sistemas de aprendizagem é uma alternativa para lidar com vários tipos de problemas, e.g. ambientes dinâmicos [9] [10], controle descentralizado [16], etc. Nós desenvolvemos e testamos uma arquitetura de aprendizagem usando agentes inteligentes para auxiliar a condução de veículos. Usualmente, os sistemas de previsão requerem uma abordagem distribuída e flexível, de forma que o sistema seja capaz de se adaptar dinamicamente às mudanças nos dados e do ambiente. Neste contexto, abordagens baseadas em agentes são adequadas para a construção de arquiteturas abertas, distribuídas, flexíveis e heterogêneas, disponibilizando uma variedade de serviços sem impor uma arquitetura centralizada e fechada *a priori*. Tal idéia é nossa principal motivação para adotar uma arquitetura baseada em agentes para um condutor eletrônico de veículos. Deste modo, o desenvolvimento incremental do piloto torna-se mais fácil [18]. Algoritmos de aprendizagem de máquina (AM) são técnicas que podem ser usadas para o desenvolvimento de sistemas incrementais [6] [13]. O agente proposto neste artigo utiliza o resultado do processo de AM, o qual inclui a extração de conhecimento a partir de bases de dados de viagens passadas [7].

2 {andre.borges, richard, allan, osmarbd, bruno.giacomet, avila, fabricio, scalabrin}@ppgia.pucpr.br

De modo pragmático, nossos esforços foram focados na implementação de técnicas de AM para extrair padrões de condução a partir de dados históricos de viagens e prover o agente com tais padrões, examinando seu comportamento na condução de um veículo de um ponto inicial *S* até um ponto final *E*, com esforço e consumo otimizados. Os padrões de condução servem como uma política de ação do agente. Uma política de ação representa o comportamento que um condutor possui, por exemplo, aumentando, mantendo ou reduzindo a velocidade [3]. Para definir uma política de ação, o agente possui um conjunto de recursos, tais como características da via e base de conhecimento contendo regras de condução de veículos. Uma boa política de ação é alcançada quando uma viagem é econômica e segura. Os padrões de condução são representados por um conjunto de regras ordenadas. A interpretação e a validação dos padrões são tarefas complexas, devido à dinâmica da aplicação e a presença de informações incompletas.

O desempenho da política de ações foi avaliado com um simulador. O comportamento do agente é confrontado com o do condutor humano usando a mesma configuração no cenário. As regras de condução do veículo foram extraídas de dados históricos usando o algoritmo RIPPER. Assim como o algoritmo C4.5 pode criar um conjunto de regras a partir de uma árvore de decisão não-podada [14][15], o algoritmo RIPPER (***R**epeated **I**ncremental **P**runing to **P**roduce **E**rror **R**eduction*), doravante chamado de JRip, constrói um conjunto de regras que modelam um conjunto de dados [5]. A seguir, técnicas baseadas em conjuntos de classificadores (Bagging [4] e Boosting [8]) são usadas para melhorar o desempenho e reduzir a taxa de erro.

Este artigo está organizado da seguinte forma: na Seção 2 é descrito conceitos básicos necessários para a compreensão do artigo e alguns dos principais trabalhos relacionados. Na Seção 3 nós descrevemos a arquitetura proposta e os resultados experimentais são apresentados na Seção 4. Finalmente, na última seção algumas conclusões são discutidas e perspectivas de trabalhos futuros são citadas.

2 Veículos Autônomos e Aprendizagem de Máquina (AM)

Técnicas de AM foram propostas para tratar problemas no qual o controle, a regularidade e a segurança são necessários. Observamos que os resultados apresentados por estes trabalhos são relacionados a veículos com características diferentes do proposto neste trabalho, considerando, por exemplo, o tamanho, capacidade de transporte, funcionalidade e peso.

Benenson et al. [1] aborda o problema de navegação autônoma de um carro-robô em um ambiente urbano com vários obstáculos dinâmicos. A técnica utilizada no tratamento do problema reside na concepção de um agente capaz de perceber e planejar ações considerando explicitamente a dinâmica do veículo e do ambiente enquanto é capaz de respeitar as restrições de segurança.

Kolski et al. [11] apresentam um sistema híbrido de navegação que combina as vantagens das abordagens existentes para a condução em ambientes estruturados (e.g., via) e ambientes não estruturados (e.g., estacionamentos). O sistema utiliza faixa de detecção visual para gerar um mapa local, que é processado por um planejador local

para orientar o veículo na pista evitando os obstáculos. Quando dirigindo em um ambiente não estruturado, o sistema emprega um mapa e um plano global para gerar uma trajetória eficiente para o objetivo desejado. O sistema híbrido é capaz de navegar um carro de passageiros para uma determinada posição, sem depender de estruturas rodoviárias ou ainda pode fazer uso de tal estrutura quando esta estiver disponível.

Bertolazzi et al. 2008 [2] conceberam um veículo autônomo de tamanho reduzido e focaram estratégias de controle baseada em um algoritmo de Controle Não-Linear (CNL). O CNL é guiado por um sistema de alto nível, para resolver uma seqüência de problemas de controle ótimo. O movimento planejado fornece uma seqüência de pontos de referência estabelecidos para o rápido controle interno até que um novo plano esteja disponível. O movimento lateral é controlado através do comando de direção e o movimento longitudinal é controlado por meio do acelerador e freio.

Além do controle e planejamento, uma maneira de resolver esses problemas é empregar técnicas de AM. Na literatura, diferentes algoritmos de aprendizagem foram propostos. C4.5 [14] é algoritmo bem sucedido, que de modo iterativo constrói uma árvore de decisão formada por um conjunto de atributos, onde as folhas apresentam o valor do atributo (nó da árvore). Em cada passo do processo de aprendizagem, a árvore é expandida com partições que geram o maior ganho de informação. Os ramos são adicionados até que todos os registros do conjunto de treinamento sejam classificados pela árvore gerada.

Para melhorar o desempenho do C4.5 ele pode ser usado um conjunto com meta-classificadores tais como os métodos Bagging e Boosting. Para cada iteração t em {1, 2, ..., T } do Bagging um conjunto de treinamento de tamanho N é gerado por amostragem a partir dos dados originais. Tais conjuntos de treinamento possuem o mesmo tamanho dos dados originais, mas em alguns casos certas instâncias podem não pertencer a certo conjunto, enquanto que em outros casos, determinadas instâncias podem aparecer mais de uma vez no mesmo conjunto. O sistema de aprendizagem gera um classificador C' para cada amostra e o classificador final C_{final} é formado agregando os T classificadores. Para classificar uma instância x, o voto de cada classificador é armazenado e finalmente a classe com a maioria dos votos é escolhida.

O método Boosting, proposto por [8], mantém um peso para cada instância x onde, quanto maior a influência da instância aprendida do classificador C_t , maior é o peso da instância. Em cada tentativa t em {1, 2, ..., T}, o peso do vetor de características é atualizado para refletir o desempenho do classificador correspondente. O erro e_t do classificador é também medido com base no peso, e consiste na somatória dos pesos das instâncias classificadas erroneamente. Se o erro de t é superior a 0,5, as tentativas terminam e T é atualizado. Se C_t classifica corretamente todas as instâncias, então e_t é 0. Caso contrário, o peso do vetor w_{t+1} é gerado pela multiplicação dos pesos das instâncias que Ct classifica corretamente pelo fator $\beta_t = e_t /(1 - e_t)$ e então, normaliza w_{t+1} de modo que a soma seja 1. O classificador final também usa o classificador aprendido por votação ponderada.

Vários estudos têm utilizado técnicas de AM para o desenvolvimento de veículos ferroviários inteligentes, cujo objetivo é melhorar a manutenção e operação dos veículos, prevendo falhas nos equipamentos. Létourneau et al. [12] discutem o mesmo tema em problemas com aeronave. Os autores usaram diferentes técnicas, como indução de

4 {andre.borges, richard, allan, osmarbd, bruno.giacomet, avila, fabricio, scalabrin}@ppgia.pucpr.br

árvores de decisão, aprendizagem baseada em instância, Naive-Bayes, Regressão e Redes Neurais.

Sammut et al. 2007 [17] aplicaram técnicas de AM para criar um conjunto de regras para robôs móveis autônomos em ambientes dinâmicos. A entrada do sistema de aprendizagem ocorreu a partir de dados históricos do ambiente. Os seguintes métodos foram utilizados: J48, Bagging e validação cruzada.

As técnicas apresentadas anteriormente foram aplicadas em problemas complexos, pois lidam com grande número de variáveis e instâncias. Além disso, estender tais técnicas para problemas do mundo-real não é uma trivial, devido à complexidade de se obter uma completa modelagem do ambiente.

3 Arquitetura de Aprendizagem Proposta

Antes de descrevermos os módulos da arquitetura proposta, nós apresentamos a descrição do veículo. O veículo é composto por marchas, representadas por um conjunto de valores discretos M = {1, 2, 3, 4, 5}, onde 1 = muito pouca força; 2 = pouca força; 3 = força média; 4 = forte; 5 = muito forte. A força de aceleração do veículo é feita empregando uma marcha.

A arquitetura proposta para um veículo autônomo (Figura 1) envolve os seguintes módulos: sensor, atuador, calculador, decisão, classificadores e refinamento. O módulo sensor captura diferentes parâmetros ao longo da viagem como: posição, velocidade e aderência. O atuador toma ações, entre elas: manter, aumentar ou reduzir a velocidade e marcha. O módulo calculador encapsula os cálculos oriundos do módulo sensor, calcula resistências e esforço trator, enriquecendo o conjunto de dados. Os módulos classificadores são responsáveis por submeter às percepções recebidas do módulo de decisão ao conjunto de regras dos classificadores, retornando a marcha a ser aplicada (Classificador A) e a o consumo da ação (Classificador B). O módulo de refinamento compreende de um conjunto de regras de segurança, que avaliam as ações resultantes do processo de classificação.

O módulo de decisão representa o comportamento principal do agente. Ele recebe valores lidos dos sensores e gera uma percepção *p*. Os valores de *p* são convertidos em um vetor *v*, onde cada posição de *v* corresponde a um atributo. O vetor *v* é submetido ao classificador *A*, que retorna uma marcha do veículo *pa*. Baseado no valor da marcha são calculados os valores da resistência, o esforço trator e o deslocamento esperado. Tais valores são inseridos no vetor *u*. Se o valor da resistência é menor que o valor do esforço trator, então o estado é gravado no arquivo de *log* e a marcha aplicada pelo atuador. Caso contrário, a marcha é aumentada. Isto acontece até que o esforço trator seja maior que a resistência gerada pelo próximo movimento. Se não é possível aumentar a velocidade, então a velocidade esperada e a atual devem ser reduzidas. O processo é repetido até que seja encontrado um valor de marcha capaz de mover o veículo de maneira satisfatória. A geração do *log* é feita para promover o processo de re-aprendizagem, que não é tratado neste artigo.

O conhecimento obtido com a aplicação da AM é representado por um conjunto de regras ordenadas. As regras ordenadas são obtidas de classificadores gerados com os

algoritmos/+métodos JRIP, JRIP+Bagging, e JRIP+Boosting, representados na arquitetura pelas camadas *Classificador A* e *Classificador B*.

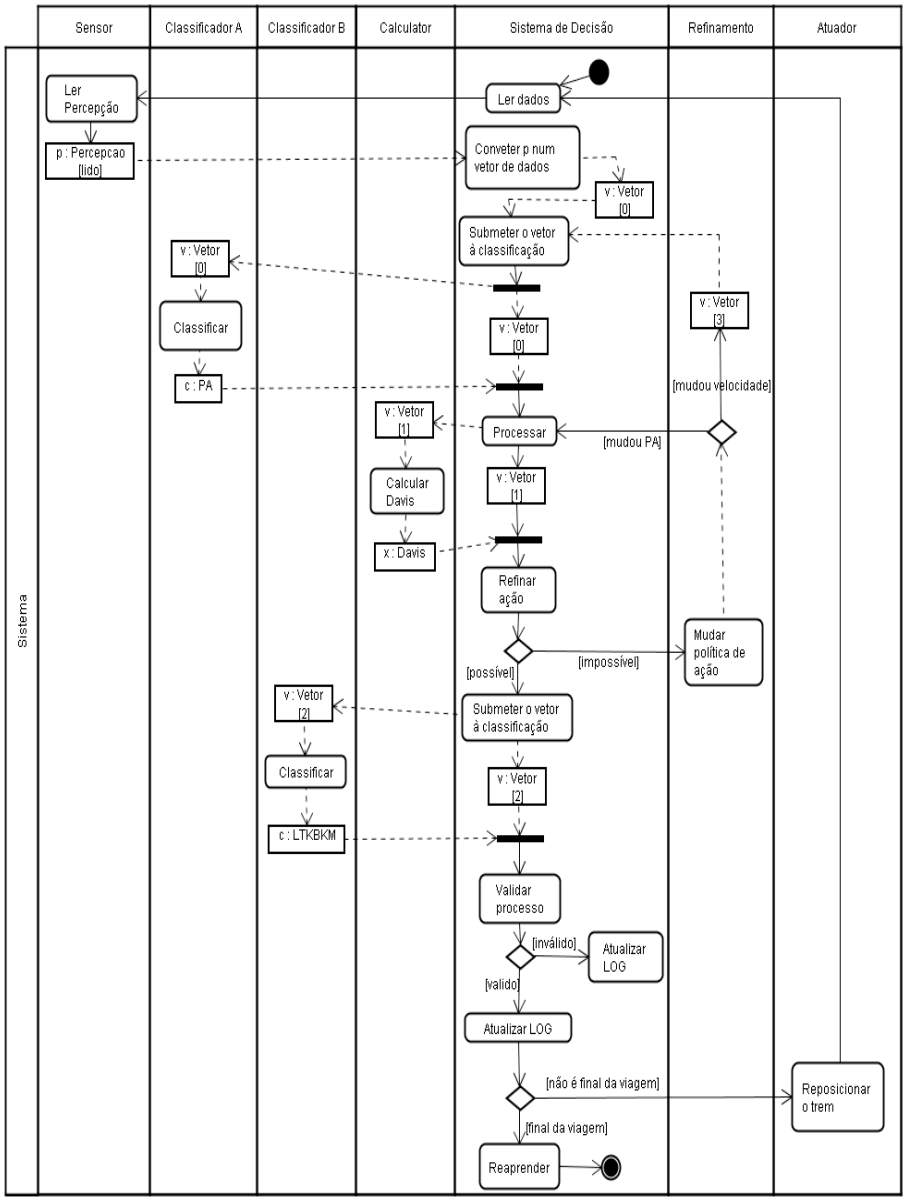


Fig. 1. Arquitetura de aprendizagem

6 {andre.borges, richard, allan, osmarbd, bruno.giacomet, avila, fabricio, scalabrin}@ppgia.pucpr.br

A figura 2 apresenta um conjunto de regras ordenadas geradas pelo algoritmo JRip. A regra R1 é lida da seguinte maneira: "SE <a velocidade do veículo é menor que 20 km/h> e <a velocidade estimada do veículo é menor que 30 km/h> então <marcha 2>".

```
R1:   (VELOCIDADE < 25) AND
      (VELOCIDADE_ESTIMADA < 30)
      => GEAR=2
R2:   (VELOCIDADE_ESTIMADA >= 22) AND
      (ANGULO_DA_CURVA <= 503) AND
      (RAMPA >= 1.33)
      => GEAR=5
[...]
```

Fig. 2. Conjunto de regras geradas pelo classificador JRIP

Portanto, tal arquitetura de aprendizagem faz uso de agentes inteligentes, características dos veículos e da via. O foco principal é a geração de regras para a condução automática de veículos, aplicando-as nos agentes para que possam aumentar, reduzir ou manter a velocidade de um veículo. Para tal, resultados foram simulados para validar a arquitetura propostas com os algoritmos/+métodos JRip, JRip+Bagging e JRip+Boosting proposta.

4 Resultados Experimentais

Nos experimentos foi usado o algoritmo JRip e também os métodos Bagging e Boosting (Ada.Boost.M1). O ambiente usado na simulação possui 60 km de via (curvas e rampas) geradas previamente por meio de levantamento cartográfico e com dados obtidos de viagens reais. Ele é definido por um conjunto de tuplas $C = (p_1, ..., p_n)$. Cada tupla representa um ponto da via e possui quatro elementos de dados *<pr, rc, vm, km>*, onde *pr* é o percentual de rampa, *rc* é a curva em metros, *vm* a velocidade máxima permitida e *km* o quilômetro. Por exemplo, a tupla <+1.0, 0, 35, 112.0> específica que o quilômetro 112 da viagem possui uma rampa ascendente com 1 por cento de inclinação, estando em reta e com velocidade máxima de 35 km/h.

De modo a medir o desempenho da solução proposta, nós comparamos a política de condução do agente com a política de condução do especialista humano. O comportamento do agente foi avaliado em duas situações: i) total de trocas de marchas; ii) velocidade praticada.

Total de trocas de Marchas

Conforme mencionado anteriormente, uma marcha gera um esforço trator que resulta em deslocamento. Uma condução é considerada boa, quando ocorrem poucas trocas de marchas. A Figura 3 ilustra a mudança de marcha gerada pelo especialista humano (dados históricos), JRip, métodos Bagging e Boosting em intervalos específicos de velocidade. Podemos observar que o especialista humano realiza troca de marchas de forma excessiva no decorrer da viagem. Isto ocorre porque as marchas são selecionadas de maneira empírica, não considerando fatores como: velocidade projetada, esforço trator, resistência, etc.

O agente que utiliza o algoritmo JRip possui a maior quantidade de troca de marchas, 325, enquanto que o condutor humano efetuou 258 trocas. Isto ocorre porque as regras geradas pelo algoritmo não possuem um mecanismo capaz de melhorar a precisão do classificador. Por outro lado, agentes que utilizam os métodos JRip+Bagging e JRip+Boosting apresentam menores quantidades de trocas de marchas. Isto ocorre porque durante a geração dos classificadores, várias amostras de dados são produzidas criando classificadores que se complementam.

O método JRip+Bagging apresentou os melhores resultados, 151 trocas. Apesar de o método JRip+Boosting ter efetuado 302 trocas e tentar se especializar na classificação de situações específicas, o seu desempenho foi inferior ao método JRip+Bagging, mostrando que espaço de tuplas de treinamento é representativo.

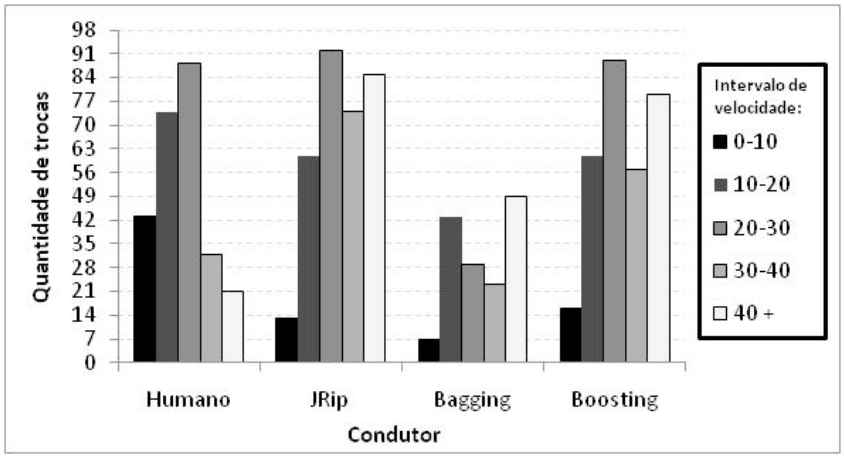


Fig. 3. Quantidade de trocas de marchas efetuadas

Variação de Velocidade

Quando uma determinada marcha é aplicada, o veículo deve ser movido a uma velocidade sem exceder a velocidade máxima permitida (VMP). A VMP é determinada

8 {andre.borges, richard, allan, osmarbd, bruno.giacomet, avila, fabricio, scalabrin}@ppgia.pucpr.br

por condições de segurança da rota para uma determinada viagem. Para haver eficiência durante o deslocamento, a velocidade deve ser constante e estar próxima da máxima, otimizando assim o consumo e o tempo de viagem. No entanto, não é tarefa trivial estabelecer de maneira precisa tal relação (marcha e velocidade) para a condução eficiente e segura em diferentes perfis de vias, mesmo para especialistas humanos. Então, os classificadores gerados buscam manter a velocidade constante e mais próxima da máxima possível.

As Figuras 5-7 mostram que as velocidades praticadas pelos classificadores são mais eficientes quando comparadas com as velocidades praticadas pelos condutores humanos (Figura 4). Isto ocorre porque o sistema tende a aumentar a velocidade, buscando a VMP. Por outro lado, o condutor humano tende a manter a velocidade abaixo da VMP. A partir dessa observação, podemos concluir que o condutor humano possui dificuldades em prever o efeito exato das ações, tornando difícil planejar as trocas de marchas em grandes intervalos de tempo, resultando na quantidade elevada de troca de marchas.

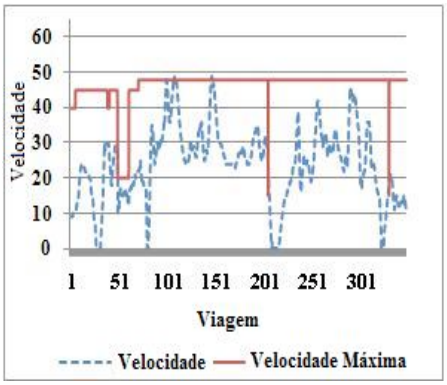


Fig. 4. Velocidades praticadas pelo condutor humano

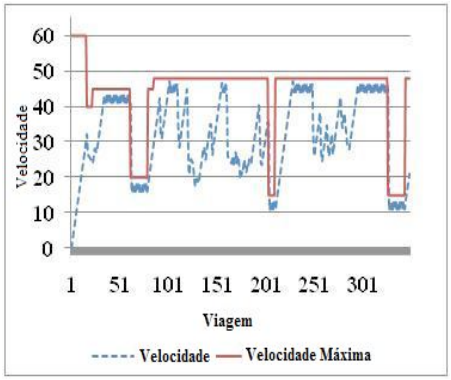
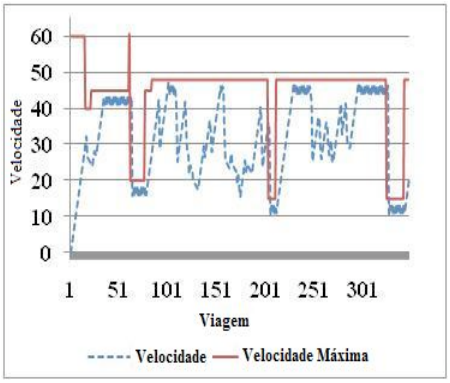


Fig. 5. Velocidades praticadas pelo JRip



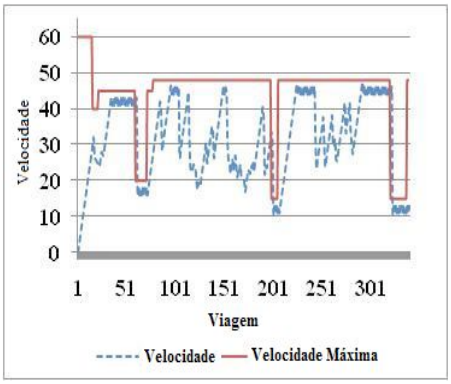


Fig. 6. Velocidades praticadas pelo método JRip+Bagging

Fig. 7. Velocidades praticadas pelo método JRip+Boosting

5 Conclusões e Discussões

Neste artigo nós apresentamos uma arquitetura de aprendizagem para agentes inteligentes capazes de auxiliar na condução automática de veículos. Um agente assistente controla a velocidade do veículo utilizando o conhecimento aprendido a partir de bases de dados. O principal esforço da pesquisa foi à indução de regras de condução de dados de viagens passadas. Foram usados os algoritmos/+métodos JRip, JRip+Bagging e JRip+Boosting, empregando dados de viagens anteriores para extrair padrões de condução segura e econômica.

O método JRip+Bagging é capaz de gerar um comportamento robusto, combinando corretamente a marcha e velocidade. Isto ocorre porque o conjunto de regras induzidas generaliza as ações de maneira adequada. No entanto, o método JRip+Boosting, pode gerar regras de condução que não são válidas de acordo com a forma de condução humana, gerando em alguns casos políticas não satisfatórias.

Apesar dos resultados iniciais serem encorajadores, estudos adicionais em outras direções são necessárias para responder algumas questões em aberto, por exemplo: i) avaliar o desempenho do agente em situações específicas de trechos, onde seria interessante verificar o desempenho dos classificadores com diferentes variações; ii) alterar o conjunto de atributos usados para a geração de classificadores, que podem torná-los menos específicos e susceptíveis à influência das condições da faixa e; iii) alterar os parâmetros dos algoritmos de aprendizagem, tais como o número de iterações. Tais indicações estão sujeitas a pesquisas futuras.

6 Agradecimentos

Nós agradecemos os revisores anônimos pelos comentários úteis e alunos do Laboratório de Agentes de Software. Essa pesquisa é financiada pela FINEP, número 3560/06, pelo projeto CNPQ Universal 470325/2009-09 e pela CAPES.

Referências

1. Benenson, R ; Petti, S, Fraichard, T., Parent, M.: Towards urban driverless vehicles. In: *International journal of vehicle autonomous systems*.vol. 6, no 1-2 pp. 4-23 (2008)
2. Bertolazzi, E., Biral, F., Bosetti, P., De Cecco, M., Oboe, R. and Zendri, F.: Development of a reduced size unmanned car. In: *Advanced Motion Control AMC 10th IEEE International Workshop on AMC*, vol. 26-28, pp. 763–770 (2008)

10 {andre.borges, richard, allan, osmarbd, bruno.giacomet, avila, fabricio, scalabrin}@ppgia.pucpr.br

3. Borges, P. A, Ribeiro, R, Ávila, C. B, Enembreck F. and Scalabrin, E. E.: A Learning Agent to Help Drive Vehicles, In: *13th International Conference on Computer Supported Cooperative Work in Design* (CSCWD'09), Santiago, Chile (2009)
4. Breiman, L.: Bagging predictors. In: *Machine learning*, vol. 24(2), pp. 123-140 (1996)
5. Cohen, W. W.: Fast Effective Rule Induction. In: *Proceedings of Twelfth International Conference on Machine Learning*, pp. 115-123 (1995)
6. Enembreck, F. and Barthès, J-P. A.: ELA - A new Approach for Learning Agents. In: *Journal Of Autonomous Agents And Multi Agent Systems*, vol. 10, n. 3, p. 215-248 (2005)
7. Fayyad, U., Shapiro, G. P. and Smyth, P.: The KDD process for extracting useful knowledge from volumes of data. In: *Communications of the ACM*, pp. 27-34 (1999)
8. Freund, Y. and Schapire, R. E.: Experiments with a new Boosting algorithm. In: *Proceedings of the 13th International Conference on Machine Learning*, pp. 148-156 (1996)
9. Isaac, A. and Sammut, C.: Goal-directed Learning to Fly. In: *Proceedings of the 20th International Conference on Machine Learning.* pp. 258-265 (2003)
10. Kaschek, R., El-Qawasmeh, E. and Tretiakov, A.: A Heuristic Understanding Model. In: *Seventh IEEE International Conference on Advanced Learning Technologies* (ICALT'07) Niigata, Japan, pp.61-63 (2007)
11. Kolski, S., Ferguson, D., Bellino, M. and Seigwart, R.: Autonomous driving in structured and unstructured environments. In: *Intelligent Vehicles Symposium*, IEEE. Tokyo, pp. 558-563 (2006)
12. Létourneau, S., Famili, F. and Matwin, S.: Data Mining for prediction of aircraft component replacement. In: *IEEE Intelligent Systems Jr., Special Issue on Data Mining*, pp. 59-66 (1999)
13. Mitchell T.: Machine Learning. New York : *McGraw-Hill* (1997)
14. Quinlan, J. R.: C4.5: *Programs for machine learning*. San Francisco: Morgan Kaufman, (1993)
15. Quinlan, J. R.: Bagging, Boosting and C4.5. In: *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pp. 725-730 (1999)
16. Ribeiro, R., Borges, A. P., Koerich, A. L., Scalabrin, E. E. and Enembreck, F.: A Strategy for Converging Dynamic Action Policies. In: *IEEE Symposium on Intelligent Agents, IEEE Symposium Series on Computational Intelligence* (SSCI'09), Nashville, USA, (2009)
17. Sammut, C., Kadous, W. and Sheh, R.: Learning to Drive Over Rough Terrain. In: *Proceedings of International Symposium on Skill Science,* (2007)
18. Scalabrin, E. E., Vandenberghe, L., Azevedo, H. and Barthès, J-P. A.: A Generic Model of Cognitive Agent to Develop Open Systems. In: *Proceedings of the 13th Brazilian Symposium on Artificial Intelligence*, vol. 1159, p. 61-70 (1996)