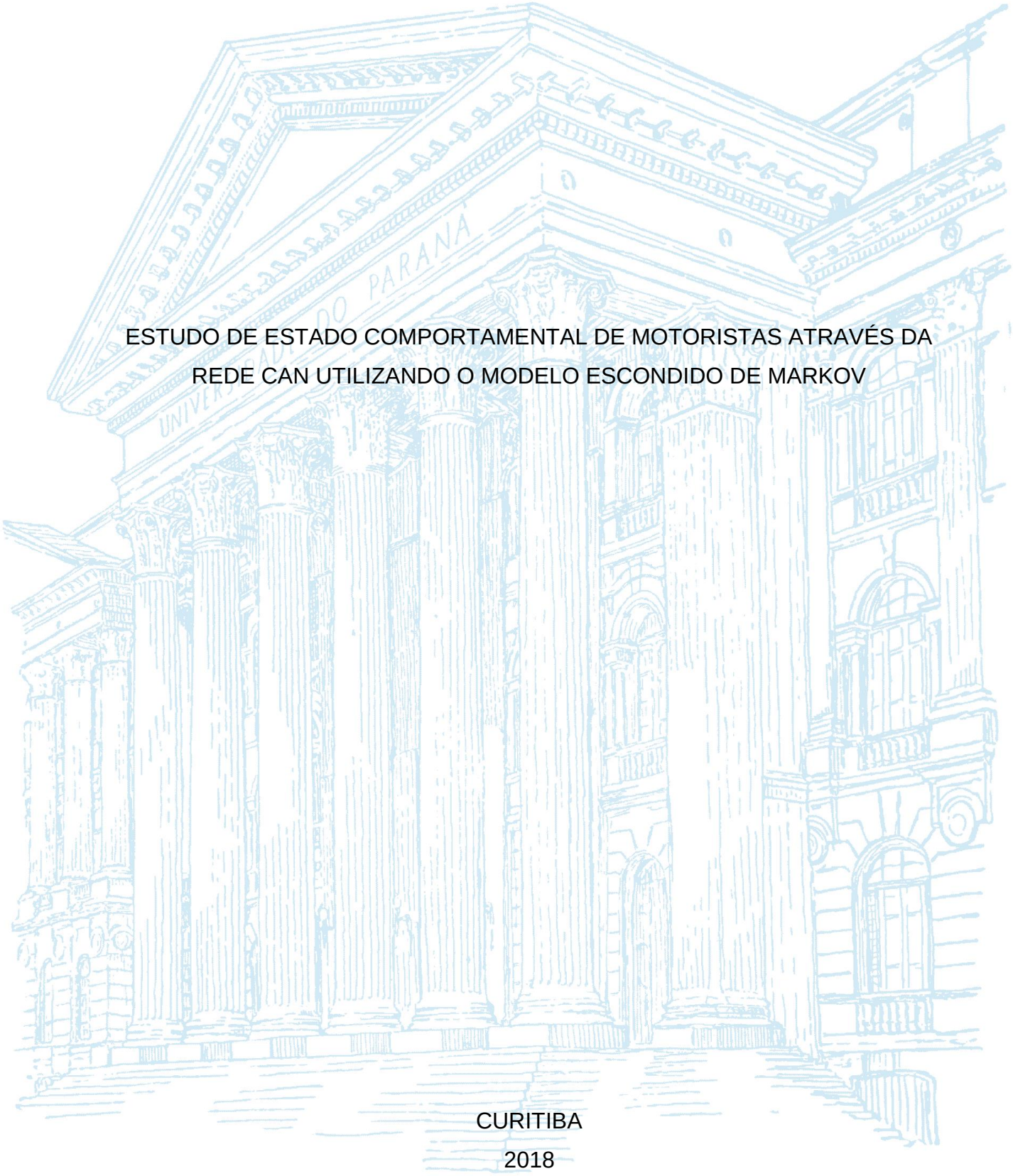
UNIVERSIDADE FEDERAL DO PARANÁ

ANDREY DE OLIVEIRA BOARÃO



ESTUDO DE ESTADO COMPORTAMENTAL DE MOTORISTAS ATRAVÉS DA REDE CAN UTILIZANDO O MODELO ESCONDIDO DE MARKOV

CURITIBA

2018

ANDREY DE OLIVEIRA BOARÃO

ESTUDO DE ESTADO COMPORTAMENTAL DE MOTORISTAS ATRAVÉS DA REDE CAN UTILIZANDO O MODELO ESCONDIDO DE MARKOV

Trabalho de conclusão de curso apresentado ao curso de Engenharia Elétrica, Setor de Tecnologia, da Universidade Federal do Paraná, como requisito à obtenção do título de Bacharel em Engenharia Elétrica.

Orientador: Prof. Dr. Carlos Marcelo Pedroso

CURITIBA

2018

TERMO DE APROVAÇÃO

ANDREY DE OLIVEIRA BOARÃO

ESTUDO DE ESTADO COMPORTAMENTAL DE MOTORISTAS ATRAVÉS DA REDE CAN UTILIZANDO O MODELO ESCONDIDO DE MARKOV

Trabalho de Conclusão de Curso apresentado ao curso de Graduação em Engenharia Elétrica Setor de Tecnologia, Universidade Federal do Paraná, como requisito à obtenção do título de Bacharel em Engenharia Elétrica.

_____

Prof. Dr. Carlos Marcelo Pedroso
Orientador – Departamento de Engenharia Elétrica, UFPR

_____

Prof. Dr. Henri Frederico Eberspacher
Departamento de Engenharia Elétrica, UFPR

_____

Prof. M. Sc. Ricardo Schumacher
Departamento de Engenharia Elétrica, UFPR

Curitiba, 04 de dezembro de 2018.

AGRADECIMENTOS

Agradeço aos meus pais, Airton e Mara, aos meus irmãos, Natacha e Guilherme e à minha noiva, Andressa, por todo o incentivo, apoio e força proporcionados.

RESUMO

Após a regularização do trabalho do motorista de veículo de transporte de carga no Brasil, exige-se que o motorista faça o controle e registro de sua jornada. Com o avanço tecnológico crescente em veículos e a grande quantidade de sensores presente neles, identificou-se a possibilidade de analisar as mensagens presentes na rede CAN, para definição do estado comportamental do motorista durante seu período de trabalho e descanso. Para tanto, foi analisado o uso de Modelos Ocultos de Markov para avaliação do estado comportamental do motorista em função do tempo. Foram feitos registros da rede CAN e o tratamento de sinais previamente selecionados, de modo a aplicar o treinamento do sistema e sua validação. No estudo, foram definidos três estados, Direção, Repouso e Abastecimento. Além disso, foram estudadas situações de taxa de amostragem reduzida e remoção de sinais com alta correlação. Para a validação do sistema, os dados obtidos foram comparados com uma referência. Comparando os resultados, identificou-se alta precisão no sistema com maior taxa de amostragem, mesmo considerando o atraso gerado entre alteração de estados. Nos outros sistemas, obteve-se uma precisão satisfatória, mas com baixo tempo de transição entre estados.

Palavras-chave: Rede CAN; Modelo Oculto de Markov; Modelo Escondido de Markov; Controle de jornada de trabalho de motorista.

ABSTRACT

After the regulation of transport vehicle driver's work in Brazil, it is demanded for the driver to control and register its journey. With the advancements in embedded vehicular electronics, it has been identified the possibility to analyze the messages within CAN bus to define driver's behavior state. Hidden Markov Model has been used to predict driver's state as a function of time. CAN bus data have been recorded and its signals were used in HMM training for its validation. In this work, it has been defined three driver's state, Driving, Rest and Refueling. It has been analyzed different sampling rate and signals. To validate the system, the results were compared with a reference file classified manually. By comparing results, it has been identified high accuracy in the system with higher sampling rate, even considering delay between state changes. In other systems, a satisfactory accuracy was obtained, but with higher delay between state transition.

Keywords: CAN bus; Hidden Markov Model; Driver journey management.

LISTA DE FIGURAS

LISTA DE GRÁFICOS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Com a busca pela inovação, atualização e a demanda do mercado, as montadoras de veículos automotores estão investindo cada vez mais em segurança, conforto, economia, luxo e entretenimento. Para tanto, os veículos atuais possuem redes de comunicação entre suas unidades eletrônicas (ECUs), as quais recebem e interpretam sinais provenientes da grande quantidade de sensores presentes nos veículos.

A partir da regulamentação da profissão do motorista não autônomo, de veículos automotores da categoria de transporte de cargas e passageiros, lei número 13.103 de 2015, exige-se que o empregado faça o controle de sua jornada e registre-a por meio físico ou eletrônico, de modo a respeitar as exigências aplicadas pela legislação trabalhista.

Este trabalho propõe a identificação do estado comportamental do motorista utilizando os sensores já presentes em um veículo automotor de transporte de carga. Tal definição será feita a partir do estudo dos sinais em função do tempo, sendo comparado com dados base obtidos a partir do comportamento do motorista em estados pré-definidos. Para estimar seu efetivo estado, será utilizado o Modelo Oculto de Markov, o qual fará uma análise dos sinais recebidos, e definirá o estado comportamental a partir de uma análise probabilística.

Foram identificados outros estudos e análises comportamentais de motoristas através de dados de veículos. Alguns destes estudos focaram na análise e definição do comportamento agressivo ou defensivo de motoristas, durante a direção do veículo. Outros, um levantamento de perfil de motorista, de modo a estimar seu comportamento em pistas e rodovias.

1.1 JUSTIFICATIVA

Este trabalho busca obter informações ocultas a partir de sinais presentes em um veículo automotor. Além da estimativa do estado comportamental, tais informações podem ser utilizadas como um estudo sobre como o motorista se comporta em relação ao caminhão durante as diferentes atividades realizadas no veículo.

1.2 OBJETIVOS GERAIS

O trabalho de conclusão de curso tem o objetivo geral de identificar o estado comportamental do motorista a partir das informações presentes nos barramentos CAN do veículo. Particularmente definindo se o mesmo encontra-se em estados típicos na rotina de trabalho de um motorista como dirigindo, abastecimento e repouso.

Para atingir tal objetivo, serão registrados comportamentos de motoristas utilizando dispositivos de análise de rede CAN e seus dados serão aplicados sobre um sistema de aprendizado, o qual interpretará, a partir do aprendizado, qual o estado comportamental que o motorista esteve em momentos específicos.

1.3 OBJETIVOS ESPECÍFICOS

Como objetivos específicos do trabalho, podem ser citados:

- Levantamento bibliográfico do material de estudo;
- Registro de dados de motoristas;
- Aplicação do sistema de aprendizado:
    - Discretização de dados;
    - Treinamento do sistema;
- Análise de resultados;
- Validação.

2 REVISÃO DE LITERATURA

O trabalho se baseia no uso de dados disponíveis na rede CAN de um caminhão para estimar o estado comportamental de um motorista. A definição, ou estimativa, do estado será feita através da aplicação do Método Oculto de Markov.

2.1 REDE CAN

Desenvolvido pela empresa alemã *Robert Bosch GmbH* em 1986, a rede CAN (*Controller Area Network)* é um protocolo de comunicação veicular serial síncrono. Focado na comunicação entre unidades de controle, as mensagens lançadas ao barramento por uma ou mais unidades são recebidas por todas as outras (*multicast)* e lidas conforme sua prioridade *(multi-mestre).* Sua comunicação é feita através de dois pares trançados, CAN HIGH (3,5V) e CAN LOW (1,5V). Ao final de cada barramento há dois terminadores, os quais são resistores de 120Ω utilizados para garantir a perfeita propagação do sinal elétrico, sendo exemplificado pelos pequenos retângulos pretos na Figura 1.

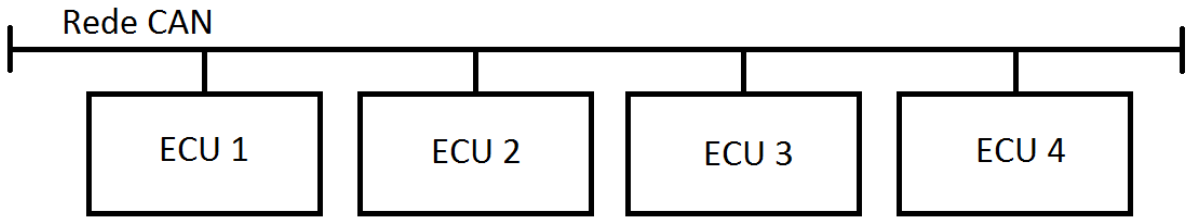


Figura 1 - Exemplo de rede CAN.
Fonte: O autor

Para que todos os módulos compreendam as mensagens contidas na rede, eles devem possuir o que é conhecido como *Dicionário de dados*, o qual é uma matriz que demonstra quais mensagens são de responsabilidade de quais módulos, a frequência de atualização de valores, intervalo de transmissão, entre outras informações. Todos os módulos de um sistema devem possuir o mesmo dicionário, o qual é implementado na unidade via *software*.

2.2 SAE J1939

Acima da camada da rede CAN, tem-se o protocolo SAE J1939, o qual é utilizado no ramo automotivo, especialmente em ônibus e caminhões. Este protocolo possui uma estrutura de mensagem com *identificador* de ECU e o campo de dados. Dentro do identificador, encontram-se os seguintes campos:

Figura 2 - Conteúdo do identificador da mensagem CAN.
Fonte: [4]

- Prioridade, um menor valor binário corresponde a uma maior prioridade.
- Página de dados, contém informações a respeito do grupo de parâmetro.
- PDU e PDU específico, formam o campo PGN (*Parameter Group Number*).
- Endereço de origem, indica o endereço da ECU que está enviando a mensagem.

Um valor de PDU abaixo de 240 em decimal, quer dizer que a mensagem é enviada especificamente para um módulo. Para valores iguais ou maiores que 240, a mensagem é um *broadcast* (também conhecido como *multicast*) onde todas as ECUs recebem a informação enviada.

Os Números de Grupos de Parâmetros, também conhecido como PGN, são grupos de mensagens/parâmetros direcionados a uma informação específica. Por exemplo, o PGN da temperatura do motor contém a informação de temperatura do líquido de arrefecimento do motor, do óleo, combustível, etc.

2.3 CADEIAS DE MARKOV

A teoria da probabilidade moderna estuda processos aleatórios onde o próximo estado do sistema depende apenas do estado atual. Em 1907, Andrei Andreyevich Markov iniciou os estudos de tais processos, que são conhecidos atualmente como processos sem memória ou Markovianos.

O modelo resultante ficou conhecido como Cadeias de Markov, os quais são métodos estocásticos que têm a propriedade de que a probabilidade de transição para um estado futuro, quando o atual é conhecido, não é alterada pelo conhecimento dos estados anteriores [7].

Supondo que tenhamos um conjunto de estados S=$\{s_1, s_2, \dots, s_n\}$. O processo inicia-se em um desses estados e varia sucessivamente de um estado para outro. Se a cadeia encontra-se em s_i e muda para outro estado s_j, a probabilidade de mudança, ou *probabilidade de transição*, é denominada P_{ij} o qual independe do estado anterior do sistema.

A Figura 3 demonstra um exemplo clássico do modelo, onde se analisa as condições meteorológicas e as probabilidades de transição entre si, tanto da troca de estado, quanto a probabilidade de manter-se no mesmo.

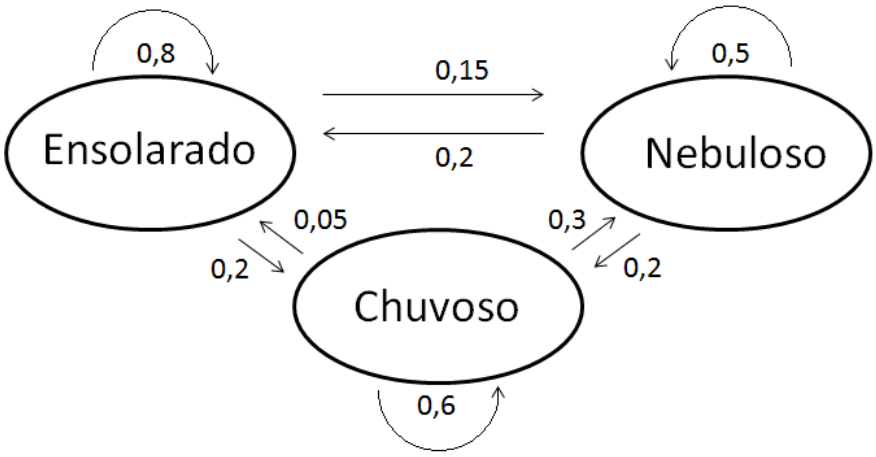


Figura 3 - Modelo de processo Markoviano.

Fonte: O autor.

Através da análise do sistema de estudo, pode-se elaborar uma matriz quadrada com as probabilidades correspondentes a cada estado e sua transição. Como exemplo, a Tabela 1 é a matriz de probabilidade de transição para o sistema apresentado na Figura 3. Onde os valores da primeira coluna representam a probabilidade de transição de diferentes estados para o estado ensolarado. Seguindo a mesma lógica, os valores das colunas conseguintes são relacionados à transição dos diferentes tipos de tempo para nebuloso e chuvoso.

		Estado de amanhã		
		Ensolarado	Nebuloso	Chuvoso
Estado atual	Ensolarado	80%	15%	5%
	Nebuloso	20%	50%	30%
	Chuvoso	20%	20%	60%

Tabela 1 - Exemplo de matriz de transição para a probabilidade do estado de amanhã baseado no estado atual.

Fonte: [9].

2.4 MODELO OCULTO DE MARKOV

Descrito em meados de 1960 por Leonard E. Baum, o Modelo Oculto de Markov, ou *Hidden Markov Model* (HMM), é uma extensão das Cadeias de Markov onde o estado é uma função probabilística de um conjunto de estados observáveis. Ou seja, o estado atual do processo não é observável, mas pode ser deduzido a partir de uma sequência de observação.

De forma a exemplificar o HMM, pode-se estender o exemplo anterior assumindo que os estados não são observáveis. Suponha que um engenheiro que trabalha em uma instalação secreta seja impedido de sair ao ar livre. A única indicação do clima que ele tem é a observação de pessoas que entram na instalação. Uma pessoa pode carregar um guarda-chuva ou uma blusa.

	Probabilidade na presença de guarda-chuva
Ensolarado	10%
Chuvoso	80%
Nebuloso	20%

Tabela 2 – Exemplo de probabilidade de emissão do Modelo Oculto de Markov.

Fonte: [9].

Na Tabela 2, é demonstrado o exemplo de probabilidade de emissão que a presença de um guarda-chuva implica sobre os estados temporais não observados. Neste caso, a probabilidade de transição entre os estados não observáveis são as mesmas das já demonstradas.

O HMM possui elementos que compõem o processo, sendo eles:

1. Um espaço de estados não observáveis *N* = $\{n_1, n_2, \dots, n_n\}$. Mesmo que não seja possível observar os estados, para muitas aplicações há algum significado físico conectado a eles.
2. Um conjunto de observações *M* = $\{m_1, m_2, \dots, m_k\}$.
3. A matriz de probabilidade de transição de estados P, *A* = $\{a_{ij}\}$.
4. Os valores de distribuição de probabilidade de emissão no estado n, *B* = $\{b_n\ (k)\}$.
5. O estado inicial de distribuição *Π* = $\{\pi_n\}$.

Dado valores apropriados para as variáveis demonstradas, gera-se a sequência de observação:

$$O = O_1 O_2 \dots O_T \qquad (1)$$

A completa especificação do HMM exige, portanto, a definição dos parâmetros N e M, além das medidas de probabilidade A, B e *Π.* Podendo ser utilizada a notação a seguir para a completa indicação dos parâmetros do modelo.

$$\lambda = (A, B, \Pi) \qquad (2)$$

Assim, pode-se dizer que, no exemplo das condições meteorológicas, as variáveis representam as seguintes situações:

- N, o número de estados possíveis para o estado meteorológico atual.
- M, a presença de um guarda-chuva ou blusa.
- A, a matriz de probabilidade de transição entre os estados: ensolarado, chuvoso e nebuloso.
- B, a matriz de probabilidade de emissão, o quanto a presença de um guarda-chuva implica na transição do estado atual.

2.4.1 Problemas e soluções do Modelo Oculto de Markov

De acordo com Rabiner [5], há três grandes problemas básicos no HMM que devem ser analisados para que o sistema seja eficientemente aplicado em situações reais:

1. Dado um HMM definido por λ e uma sequência de observações B={$b_1, b_2, \ldots, b_k$}, determinar a probabilidade das observações terem sido geradas pelo modelo, ou seja P{B|λ}.
2. Dado um HMM definido por λ e uma sequência de observações B={$b_1, b_2, \ldots, b_k$}, determinar o estado mais provável do sistema.
3. Dado um HMM definido por λ e uma sequência de observações B={$b_1, b_2, \ldots, b_k$}, determinar os parâmetros {A, B, Π} *para maximizar P{B|λ}*.

Tais problemas são resolvidos pelos seguintes algoritmos:

1. Neste caso, Rabiner recomenda o uso do algoritmo ***Forward***. O qual armazena as probabilidades de mudança dos estados através da observação da sequência de observações.
2. Há diversas formas de encontrar o estado mais provável do sistema, no entanto, recomenda-se o ***Algoritmo de Viterbi***. Sendo o critério mais utilizado, este algoritmo leva em conta casos de onde haja probabilidades de transição iguais à zero, além disso, busca pela melhor sequência (caminho) de estados dada a sequência observada.
3. *O* ***algoritmo de Baum-Welch***, o qual é um processo iterativo para encontro de máxima probabilidade do sistema a partir da sequência de observações.

A seguir são demonstradas as deduções dos problemas de acordo com Rabiner [5]. Tais deduções são utilizadas apenas como referência ao trabalho, não sendo necessário o entendimento delas para progressão na leitura do capítulo seguinte.

2.4.1.1 Solução do problema 1

O cálculo da probabilidade de uma sequência de observação, como demonstrado em (1), a partir do modelo (2), ou P(O|λ), pode ser feito de maneira direta, enumerando todas as possibilidades de transição de estado, ou através do uso de algoritmos.

Considerando que se tenha a seguinte sequência de estados:

$$Q = q_1 q_2 \dots q_T \tag{3}$$

Onde q_1 é o estado inicial, e T é o número de estados analisados. A probabilidade da sequência de observações O para a sequência de estados (3) é

$$\mathrm{P}(\mathrm{O}|\mathrm{Q},\lambda) = \prod_{t=1}^{T} \mathrm{P}(O_t|Q_t,\lambda) \tag{4a}$$

Ou seja,

$$\mathrm{P}(\mathrm{O}|\mathrm{Q},\lambda) = b_{q_1}(O_1).b_{q_2}(O_2) \dots b_{q_t}(O_t) \tag{4b}$$

A probabilidade de sequência de estados Q é escrito como:

$$\mathrm{P}(\mathrm{Q}|\lambda) = \pi_{q_1} a_{q_1 q_2} a_{q_2 q_3} \dots a_{q_{T-1} q_T} \tag{5}$$

E a probabilidade de ocorrência de Q e O simultaneamente é

$$\mathrm{P}(\mathrm{O},\mathrm{Q}|\lambda) = \mathrm{P}(\mathrm{O}|\mathrm{Q},\lambda)\mathrm{P}(\mathrm{Q},\lambda) \tag{6}$$

Assim, a probabilidade de O é obtida através do somatório de (6) em todas as possíveis sequências de estado q, ou seja:

$$\mathrm{P}(\mathrm{O}|\lambda) = \sum_{Q} \mathrm{P}(\mathrm{O}|\mathrm{Q},\lambda)\mathrm{P}(\mathrm{Q}|\lambda) \tag{7}$$

$$\mathrm{P}(\mathrm{O}|\lambda) = \sum_{q_1,q_2,\ldots,q_T} \pi_{q_1} b_{q_1}(O_1) a_{q_1 q_2} b_{q_2}(O_2) \ldots a_{q_{T-1} q_T} b_{q_T}(Q_T) \qquad (8)$$

A interpretação das equações (7) e (8), nos mostra o seguinte: No tempo inicial (T=1), tem-se o estado q_1 com a probabilidade π_{q_1}, a qual gera o símbolo O_1 com a probabilidade $b_1(O_1)$. O tempo, então, passa para t=2, onde a probabilidade de transição do estado q_1 para o estado q_2 é $a_{q_1 q_2}$, gerando O_2 com a probabilidade $b_2(O_2)$. Esta lógica ocorre até o estado O_T.

O cálculo de $\mathrm{P}(\mathrm{O}|\lambda)$, utilizando esta definição direta, envolve operações na ordem 2T*$N^T - 1$, onde N são os estados e T as posições no tempo. Ou seja, o cálculo se torna possível, mas inviável até mesmo para pequenos valores de N e T.

De forma a calcular a probabilidade de forma mais eficiente, utiliza-se o algoritmo *forward-backward.* Sendo apenas a primeira parte (*forward)* necessária para o cálculo.

Considere a variável *forward* $\alpha_{\mathrm{t}}(\mathrm{i})$, definida como:

$$\alpha_{\mathrm{t}}(\mathrm{i}) = \mathrm{P}(O_1 O_2 \ldots O_t, q_t = S_i|\lambda) \qquad (9)$$

A resolução de $\alpha_{\mathrm{t}}(\mathrm{i})$ é feita por três passos:

1) Inicialização: A qual inicia a probabilidade *forward* como a junção de probabilidades entre o estado S_1 e a observação inicial O_1. Representado pela equação (10).

$$\alpha_1(\mathrm{i}) = \pi_i b_i(O_1) \ , \qquad 1 \leq \mathrm{i} \leq \mathrm{N} \qquad (10)$$

2) Indução: Ilustrado pela equação (11). Visto que $\alpha_{\mathrm{t}}(\mathrm{i})$ é a probabilidade de junção do evento em que as observações parciais $O_1 O_2 \ldots O_t$ são observadas e o estado no tempo t é S_i. Ao multiplicar $\alpha_t(i) a_{ij}$, calcula-se a transição do estado S_i para S_j no tempo t+1. Somando o produto por todas as N possibilidades, resulta na probabilidade do estado S_j no tempo t+1 acompanhado das observações parciais anteriores. Assim,

$\alpha_{t+1}(j)$ é obtido através da multiplicação do valor somado obtido por $b_j(O_{t+1})$.

$$\alpha_{t+1}(j) = [\sum_{i=1}^{N} \alpha_t(i)a_{ij}]\, b_j(O_{t+1}),\ \ 1 \leq t \leq T-1 \qquad 1 \leq j \leq N \tag{11}$$

3) Terminação: O último passo é definido pela soma das variáveis *forward* finais $\alpha_T(i)$.

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i) \tag{12}$$

O algoritmo *forward* requer cálculos de ordem N^2T, exigindo menor esforço computacional se comparado ao método demonstrado anteriormente.

Analogicamente, tem-se o algoritmo *backward* que é o reverso do algoritmo *forward*, definido de acordo com a equação (13).

$$\beta_t(i) = P(o_{t+1}o_{t+2} \ldots o_T | q_t = s_i, \lambda) \tag{13}$$

Sendo $\beta_t(i)$ a probabilidade de observação parcial da sequência de t+1 à T, iniciando no estado s_i.

2.4.1.2 A solução do problema 2

Existem diversas formas de se encontrar a sequência ótima de estados de um sistema. No entanto, como demonstrado por Blunsom [12] e Rabiner [5], o algoritmo mais recomendado é o algoritmo de *Viterbi*..

Sendo Q a sequência de estados e O a sequência observável, a maior probabilidade em um único caminho é dada por $\delta_t(i)$, no tempo t, contabilizando as primeiras t observações e acabando no estado s_i.

$$\delta_t(i) = \max_{q_1, q_2 \ldots q_{t-1}} P(q_1 q_2 \ldots q_t = s_i,\ O_1\ O_2 \ldots\ O_t |\ \lambda) \tag{14}$$

Por indução têm-se

$$\delta_{t+1}(i) = [\max_{i} \delta_t(i) a_{ij}] b_j(O_{t+1}) \quad (15)$$

Sendo ψ_t um vetor de armazenamento de estados maximizados, guardando nas posições t o índice j do estado S_j que maximiza a sequência para o próximo estado.

O algoritmo de *Viterbi* é dividido em:

1. Inicialização:

$$\delta_1(i) = \pi_i b_i(O_1) \qquad 1 \leq i \leq N \quad (16)$$

$$\psi_t(j) = 0 \quad (17)$$

2. Indução:

$$\delta_t(j) = \max_{1 \leq i \leq N} \left[\delta_{t-1}(i) a_{ij}\right] b_j(O_t) \qquad 2 \leq t \leq T, \quad 1 \leq j \leq N \quad (18)$$

$$\psi_t(j) = \operatorname{argmax}_{1 \leq i \leq N} \left[\delta_{t-1}(i) a_{ij}\right], \qquad 2 \leq t \leq T, \quad 1 \leq j \leq N \quad (19)$$

3. Finalização:

$$P^* = \max_{1 \leq k \leq N} [\delta_T(k)] \quad (20)$$

$$q_T^* = \operatorname{argmax}_{1 \leq k \leq N} [\delta_T(k)] \quad (21)$$

4. Recriação do caminho (sequência de estado):

$$q_t^* = \psi_{t+1}(q_{t+1}^*), \qquad t = T-1, T-2, \ldots, 1 \quad (22)$$

Este algoritmo tem certa semelhança com o *forward*, sendo sua maior diferença no passo de indução, onde há o uso dos valores maximizados.

2.4.1.3 A solução do problema 3

Para determinar um método que ajuste o modelo de parâmetros $\lambda = (A, B, \Pi)$ de modo a maximizar a probabilidade da sequência de observação P(O|λ), dado um modelo, recomenda-se o algoritmo *Bawn-Welch*, o qual faz atualizações e melhoria de forma iterativa nos parâmetros do HMM.

Considere a seguinte definição:

$$\xi_t(i,j) = P(q_t = S_i, q_{t+1} = S_j | O, \lambda) \quad (23)$$

Dado a sequência O de treinamento, a variável $\xi_t(i,j)$ é a probabilidade de estar no estado S_i em t e no estado S_j em t+1. Com os algoritmos *forward* e *backward* já demonstrados, pode-se escrever a equação (23) na forma

$$\xi_t(i,j) = \frac{\alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{P(O|\lambda)} = \frac{\alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)} \quad (24)$$

Definindo $\gamma_t(i)$ como

$$\gamma_t(i) = P(q_t = S_i | O, \lambda) \quad (25)$$

Ou seja, a probabilidade de estar no estado S_i no tempo t, dado um modelo e a sequência de observação. Pode-se expressar tal equação utilizando os algoritmos *forward* e *backward* da seguinte forma

$$\gamma_t(i) = \frac{\alpha_t(i) \beta_t(i)}{P(O|\lambda)} = \frac{\alpha_t(i) \beta_t(i)}{\sum_{i=1}^{N} \alpha_t(i) \beta_t(i)} \quad (25)$$

A qual pode ser relacionada com $\xi_t(i,j)$ da seguinte maneira

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i,j) \quad (26)$$

Somando $\gamma_t(i)$ sobre o tempo T, obtêm-se a estimativa do número de vezes que é atingido o estado S_i. O número de transições sobre S_i, deve-se levar o somatório até o tempo T-1. Assim, pode-se definir formalmente

$$\sum_{t=1}^{T-1} \gamma_t(i) = número\ esperado\ de\ transições\ de\ S_i \quad (27)$$

$$\sum_{t=1}^{T-1} \xi_t(i,j) = número\ esperado\ de\ transições\ de\ S_i\ para\ S_j \quad (28)$$

Através das fórmulas acima, pode-se reestimar os parâmetros de um HMM para as seguintes variáveis:

$$\bar{\pi} = Frequência\ esperada\ no\ estado\ S_i\ no\ tempo\ t = 1 = \gamma_t(i) \quad (29)$$

$$\overline{a_{ij}} = \frac{Número\ esperado\ de\ transições\ de\ S_i\ para\ S_j}{Número\ esperado\ de\ transições\ de\ S_i} \quad (30)$$

$$\overline{b_j}(k) = \frac{Número\ esperado\ de\ transições\ no\ estado\ j\ e\ observação\ v_k}{Número\ esperado\ no\ estado\ j} \quad (31)$$

Assim, definindo o modelo atual como $\lambda = (A, B, \Pi)$ e utilizando as equações (29)-(31) para criar um novo modelo $\bar{\lambda} = (\bar{A}, \bar{B}, \bar{\Pi})$, Baum e Baker provam que:

1. O modelo inicial é um ponto crítico, onde a função de probabilidade é um ponto de máximo, ou seja $\lambda = \overline{\lambda}$;
2. O novo modelo $\overline{\lambda}$ é mais provável que o antigo, sendo esse o novo modelo uma sequência mais provável de observação.

O processo é executado até o encontro do ponto de limite, onde $\lambda = \overline{\lambda}$.

3 MATERIAIS E MÉTODOS

Nesta sessão são apresentados os materiais necessários para elaboração dos experimentos, assim como a definição, tratamento e interpretação dos sinais adquiridos e sua aplicação no sistema de aprendizado.

3.1 MONITORAMENTO DA REDE CAN

Para análise das redes CAN disponíveis no veículo, utilizou-se a *CANcaseXL* da *Vector Informatik.* Este *hardware* é uma interface USB com duas entradas DB9 (dois canais), as quais possuem canais CAN independentes. Assim, para leitura do barramento, deve-se ramificar o par trançado da CAN do veículo e colocar um conector DB9 para integração com a interface. Com ela, é possível ler e transmitir mensagens CAN 11bits e 29bits, além de leitura e geração de códigos de falha.

O programa utilizado para a interpretação dos dados lidos é o CANalyzer 9.0 também da *Vector Informatik.* Este programa exibe as mensagens CAN do veículo de maneira que seja possível analisar o que se passa no barramento, como os identificadores e seus bits sendo alterados de acordo com a informação presente no veículo. A Figura 4 ilustra o uso do programa para leitura das mensagens, onde a coluna “Data” exibe na cor preta os hexadecimais que sofreram recente alteração nos pacotes.

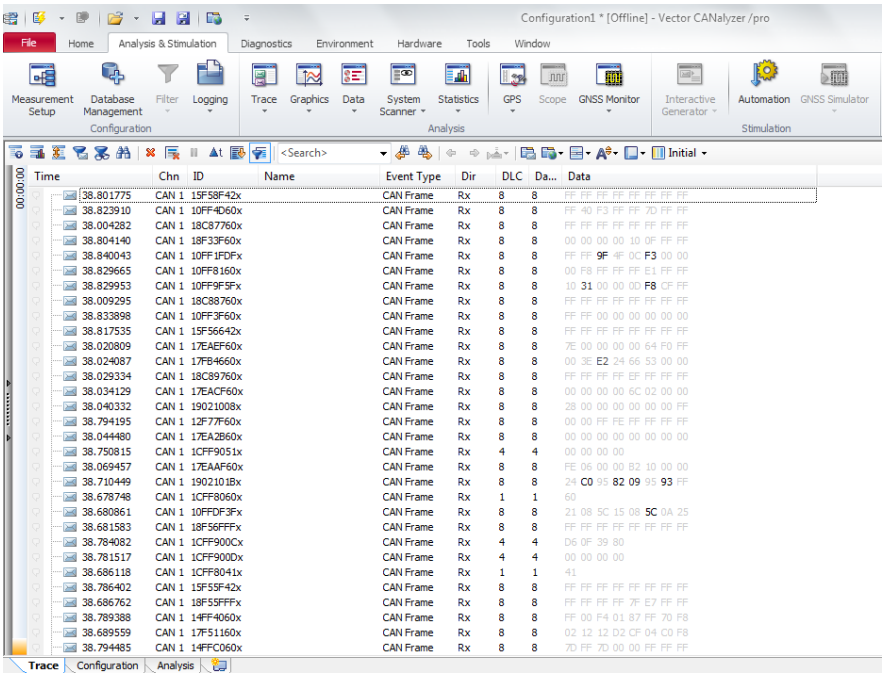


Figura 4 - Exemplo de funcionamento do programa CANalyzer.

Fonte: O autor.

Para possível compreensão do barramento lido, é necessária a aplicação de uma data-base proprietária do veículo disponibilizada pela montadora, que contém a informação dos identificadores como nome da mensagem, os sinais presentes neles e seus valores correspondentes.

Para análise do comportamento pela rede CAN, escolheu-se o barramento que possui mensagens tanto de interface com o motorista quanto informações do motor e sensores. Após análise dos sinais disponíveis no barramento por meio da data-base, foram selecionados aqueles que se supõem ser de grande impacto sobre o comportamento do motorista, sendo eles:

- Porcentagem da pressão sobre o pedal do acelerador;
- Porcentagem da pressão sobre o pedal do freio;
- Estado da tranca da porta;
- Nível de combustível;
- Volume do rádio;
- Posição da chave de ignição;
- Velocidade.

Além da definição dos sinais, deve-se analisar sua posição em relação à mensagem. Para tanto, utilizou-se novamente das informações presentes na data-base, a qual informa o *startbit* de cada sinal sobre a mensagem e seu tamanho. Deste modo, ao executar uma gravação da rede CAN, pode-se identificar na mensagem a variação dos sinais presentes no arquivo de registro em função do tempo e seu conteúdo.

Ao registrar um arquivo de *log* como exemplo para análise, identificou-se a possibilidade de exportar o arquivo como uma tabela no formato ASCII, o qual torna fácil a compreensão do texto presente no documento gerado. A Tabela 3 contém um exemplo de como é formada a tabela no arquivo de *log*.

Tempo	Canal	ID Hex	Direção	DLC	Data field								Comprimento	Contagem de bits	ID Dec
					0	1	2	3	4	5	6	7			
0.000000	1	FFFFFFFFx	Rx	d 8	FF	FF	FF	FF	FF	FF	FF	FF	Length = 273910	BitCount = 141	ID = 4294967295x

Tabela 3 - Estrutura da mensagem CAN exportada pelo CANalyzer em ASCII.

Tal tabela contém informações a respeito do momento de leitura, como o tempo, o identificador e o *data field*, o qual é o conteúdo dos sinais. Com essas três informações, pode-se interpretar o momento em que o sinal foi variado no tempo.

3.2 AQUISIÇÃO DE DADOS

Após a instrumentação do veículo com o *CANalyzer*, iniciou-se a aquisição de dados. Para tanto, solicitou-se que um motorista qualificado executasse a direção do veículo durante, aproximadamente, duas horas. Foram executados os seguintes registros para treinamento:

- Repouso: 447 segundos;
- Direção do ponto de início ao posto de abastecimento: 385 segundos;
- Abastecimento: 815 segundos;
- Direção do posto de abastecimento ao posto de carga: 411 segundos;
- Direção do posto de carga ao ponto de início: 408 segundos;
- Repouso: 665 segundos.

Em outro momento, executou-se mais uma quantidade de medições:

- Do ponto de início ao posto de abastecimento: 614 segundos;
- Abastecimento: 776 segundos;
- Do ponto de abastecimento ao ponto de início: 1386 segundos;
- Repouso: 489 segundos.

De forma a validar o sistema, registrou-se uma viagem contendo 7553 segundos de validação.

3.3 TRATAMENTO DE DADOS

Para que a informação presente no veículo seja interpretada com melhor precisão pelo software de análise estatística, é necessário que o registro do barramento *CAN* seja tratado de acordo com a particularidade de cada sinal. Assim, com a aquisição de um registro exemplo, foram elaborados *scripts* para tratamento dos sinais:

1. Filtragem por mensagem:

   Após a exportação do documento puro vindo do *CANalyzer*, obtém-se uma tabela onde os sinais são listados em função do tempo, como já demonstrado. Aplicou-se um filtro sobre as mensagens, de modo a gerar um arquivo com apenas a mensagem pertinente ao estudo.

2. Seleção de sinal:

   Utilizando o arquivo contendo a mensagem filtrada e, com a informação da *data field* contida na data-base da *CAN*, foi selecionada a coluna correspondente ao sinal a ser analisado. Aplicou-se, portanto, outro filtro sobre o arquivo de modo a obter apenas a coluna referente ao sinal.

3. Pré-processamento dos dados:

   Devido ao fato de as mensagens conterem distintas taxas de amostragem, foi aplicado uma média móvel exponencial sobre os dados como demonstrado a seguir:

$$MME = (V_i - V_{i-1}) * \alpha + V_{i-1} \quad (32)$$

Onde V_i é o valor medido e *α* a variável que define o peso do sinal mais recente sobre a média. Para obtenção do valor de *α,* utilizou-se novamente a data-base da rede de estudo para analisar qual o ciclo de cada mensagem, de modo a aplicar um valor de alfa de acordo com a periodicidade de cada uma. Assim, definiu-se um valor de *α* para cada taxa de amostragem, sendo ele medido de maneira empírica até que a média exponencial mantenha a variação do sinal por um segundo. A tabela a seguir demonstra sobre quais sinais aplicou-se a média móvel exponencial e seus respectivos valores de *α*:

Sinal	Valor de α
Acelerador	0,18
Combustível	Não aplicado
Freio	0,18
Porta	0,9
Radio volume	0,7
Ignição	0,18
Velocidade	0,5

Tabela 4 - Valores de *α* para cada sinal analisado.

Fonte: O autor.

4. Discretização e amostragem:

Através da interpretação dos sinais avaliados, aplicou-se a discretização do sinal onde, novamente, utilizou-se da data-base para interpretação da relação entre sinal e evento. Para a discretização, foi definido um intervalo de valores entre 0 e 2, de modo a obter, no máximo, três possíveis estados para cada sinal. Além disso, para os sinais com alta taxa de amostragem, foi aplicado um filtro de modo a adquirir amostras do sinal de acordo com sua variação no tempo de maneira inversamente proporcional a sua taxa de amostragem. Ou seja, quanto maior sua taxa, menor o número de amostras adquiridas, de modo a igualar a taxa de amostragem de todos os sinais.

Cada sinal foi tratado da seguinte maneira:

- **Acelerador:** O sinal do acelerador é definido pela pressão que o motorista aplica sobre o pedal do mesmo. Deste modo, definiu-se o valor 0 para o caso onde não haja pressão, e 2 para valores de pressão acima de zero. Aplicou-se um filtro de amostragem de 1 amostra a cada 100 leituras.
- **Combustível:** O sinal do combustível é definido pela altura que o medidor de nível está em relação a sua referência. Assim sendo, definiu-se o valor 0 para casos onde o nível atual é idêntico ao lido anteriormente, e o valor 2 para casos onde o nível medido é diferente do anterior.
- **Freio:** O sinal de freio comporta-se de maneira muito similar ao acelerador. Desta forma, aplicou-se a mesma lógica em sua implementação. Aplicou-se um filtro de amostragem de 1 amostra a cada 100 leituras.
- **Estado da tranca da porta:** Este sinal é composto por um conjunto de sinais. Desta forma, aplicou-se um intervalo sobre os valores máximos e mínimos em que cada estado da porta encontra-se. Assim, na situação onde ambas as portas estão abertas, o valor é igual a 0. O valor é igual a 1 no caso de apenas uma porta aberta, e 2 para ambas trancadas. Aplicou-se um filtro de amostragem de 1 amostra a cada 2 leituras.
- **Intensidade do som:** De uma maneira empírica, definiu-se três intervalos de intensidade do som, sendo baixo, médio e alto. Assim, o valor 0 encontra-se no intervalo de baixa intensidade, 1 para média e 2 para alta. Aplicou-se um filtro de amostragem de 1 amostra a cada 5 leituras.
- **Chave de ignição:** Este sinal interpreta a posição da chave de ignição. Sendo, o valor 0 definido para a chave de ignição fora do miolo, 1 para a posição conhecida como “acessórios”, onde o rádio mantém-se ligado em conjunto com a maioria das ECUs, e o valor 2 para motor em estado de funcionamento. Aplicou-se um filtro de amostragem de 1 amostra a cada 100 leituras.

- **Velocidade:** Através do cálculo da velocidade da roda, definiu-se o valor 0 para estado de roda parada e 2 para velocidade acima de zero. Aplicou-se um filtro de amostragem de 1 amostra a cada 33 leituras.

Todos os códigos aplicados estão presentes no anexo I.

Para análise, foram estudados sistemas de taxas de amostragem diferentes, de forma a verificar a alteração do valor da probabilidade de transição de estados. Para tanto, multiplicou-se a quantidade de leituras por amostra pelo valor da nova taxa de amostragem.

Além disso, analisou-se a correlação entre os sinais de estudo, de forma a verificar o impacto da remoção de sinais que possuem alta correlação entre si. A Tabela 5 contém os valores obtidos.

	Acelerador	Combustivel	Freio	Porta	Volume	Ignicão	Movimento
Acelerador	-	-0,109	-0,112	-0,112	0,054	0,318	0,857
Combustivel	-0,109	-	-0,044	-0,079	0,012	0,103	-0,115
Freio	-0,112	-0,044	-	-0,093	0,063	0,116	0,133
Porta	-0,112	-0,079	-0,093	-	-0,434	-0,547	-0,122
Volume	0,054	0,012	0,063	-0,434	-	0,091	0,057
Ignicão	0,318	0,103	0,116	-0,547	0,091	-	0,337
Movimento	0,857	-0,115	0,133	-0,122	0,057	0,337	-

Tabela 5 - Correlação entre sinais de estudo.

Fonte: O autor.

Assim, outra situação de estudo foi executada onde os sinais de “Movimento” e “Porta” foram removidos, devido sua alta correlação com outros sinais.

Após o tratamento dos sinais, os valores obtidos foram inseridos em uma tabela onde cada coluna corresponde aos sinais de estudo. Fez-se o uso de um *software* de manipulação de dados de forma a implementar a lógica por trás do HMM no sistema de estudo.

3.4 TREINAMENTO E VALIDAÇÃO

Para aplicação do HMM, utilizou-se de um *software* voltado para manipulação de dados estatísticos e análise de gráficos. Chamado de *R,* este *software* é uma ferramenta de uso livre desenvolvida com base no *software S* desenvolvido por John Chambers da *Bell Laboratories [18]*.

Executou-se a instalação de pacotes de análise de dados no *software* R, os quais aplicam médias móveis, análise gráfica e o próprio HMM. Os pacotes utilizados são:

- DepmixS4 - é um pacote desenvolvido por Ingmar Visser e Maarten Speekenbrink adaptado para aplicação do Modelo Oculto de Markov sobre dados categóricos mistos e contínuos (séries temporais), conhecido como “modelos de mistura dependente”.
- QCC - desenvolvido por Luca Scrucca, Greg Snow e Peter Bloomfield, este pacote faz aplicação da média móvel exponencial sobre os dados.
- Cluster – desenvolvido por Martin Maechler, Peter Rousseeuw, Anja Struyf, Mia Hubert, Kurt Hornik, Matthias Studer, Pierre Roudier, Juan Gonzalez e Kamil Kozlowski, este pacote executa a clusterização dos dados inseridos para análise de grupos.

De modo a definir a quantidade de possíveis estados do motorista identificados pelo sistema, aplicou-se a técnica de clusterização, uma função já presente no *software R.* Esta técnica faz a análise dos dados previamente inseridos no sistema e agrupa-os de acordo com sua similaridade. Para tanto, aplicou-se os dados obtidos para validação do sistema, após a execução de seu tratamento. A clusterização tem maior efetividade em dados normalizados, portanto, a aplicação dos dados já tratados aperfeiçoa o resultado.

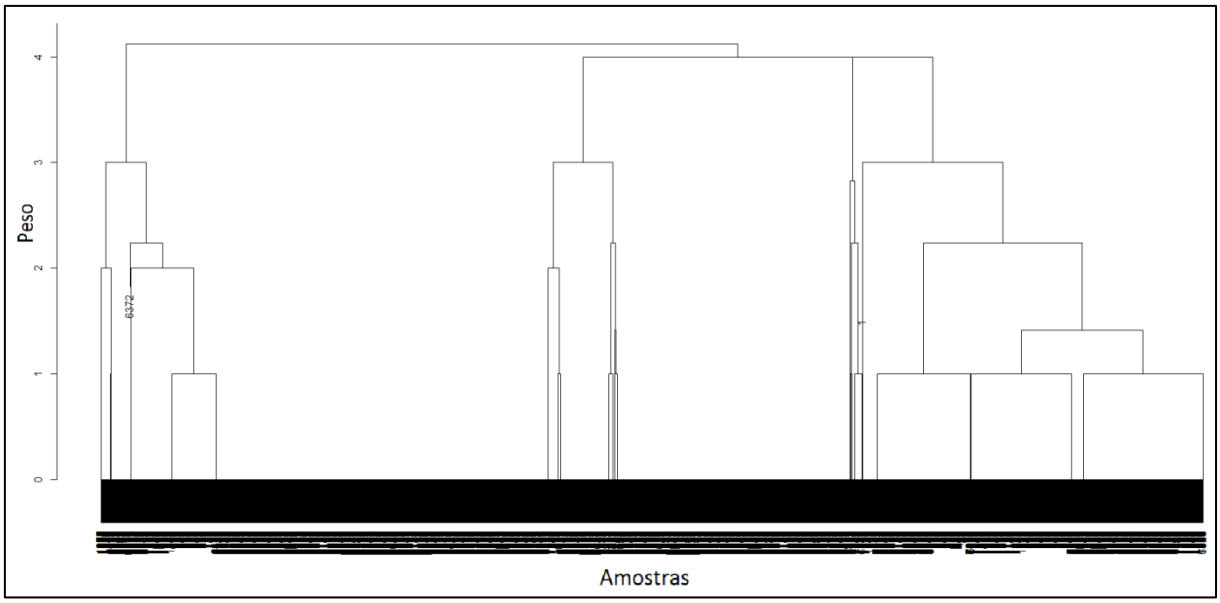


Gráfico 1 - Gráfico de análise de agrupamento dos dados de validação do sistema.

Fonte: O autor.

Através do resultado obtido da clusterização, Gráfico 1, optou-se por limitar o sistema em três possíveis estados.

Com todas as informações obtidas, pode-se fazer a aplicação do HMM seguindo os seguintes passos no *R:*

1. Leitura de tabela de sinais tratados para aprendizado, registrando tais dados como uma variável;
2. Aplicação de média móvel exponencial sobre cada coluna presente na tabela;
3. Aplicação da função *depmix,* especificando a distribuição observada. Ou seja, contendo os sinais, seu tipo (contínuo ou discreto) e o número de estados esperados;
4. Aplicação da função *fit*, que realiza o treinamento da HMM, obtendo λ.

Assim, já se obtém o modelo contendo suas probabilidades de transição e emissão. Para que o modelo seja validado, foram executados os seguintes passos:

1. Leitura de tabela de sinais tratados para validação;
2. Aplicação de média móvel exponencial sobre cada coluna;
3. Aplicação da função *depmix*;

4. Função *setpars*, de modo a aplicar os parâmetros do modelo de aprendizado;
5. Função *viterbi* para determinar o estado mais provável a partir de λ e um novo conjunto de observações B'.

Os dados obtidos foram analisados e comparados com o estado efetivo do motorista.

4 RESULTADOS

Os resultados apresentados são divididos de acordo com sua taxa de amostragem e quantidade de sinais presentes.

4.1 PRIMEIRA ANÁLISE

Para a primeira análise, utilizou-se a taxa de amostragem mínima obtida pela leitura do veículo, sendo de 1 segundo. Os sinais de estudo desta análise são: Acelerador, combustível, freio, porta, volume, chave de ignição, e movimento.

Ao aplicar o sistema de treinamento, obtêm-se as probabilidades de transição e emissão dos estados.

	Repouso	Abastecimento	Direção
Repouso	0,996	0,001	0,003
Abastecimento	0,004	0,996	0,000
Direção	0,002	0,000	0,998

Tabela 6 - Matriz de probabilidade de transição de estado da primeira análise.

Fonte: O autor.

	Aceleração	Combustível	Freio	Porta	Volume	Ignição	Movimento
Repouso	0,078	0,201	0,020	0,703	0,599	0,396	0,087
Abastecimento	0,005	0,361	0,001	0,436	0,944	0,006	0,006
Dirigindo	0,524	0,015	0,175	0,425	0,589	0,192	0,581

Tabela 7 - Matriz de probabilidade de emissão por estado da primeira análise.

Fonte: O autor.

Analisando a Tabela 6, identifica-se que há situações onde a probabilidade de transição iguala-se a zero. Isso de dá devido ao fato de o *software* utilizado truncar a probabilidade na terceira casa decimal. Nota-se, também, que a probabilidade de o motorista manter-se no mesmo estado é muito alta em relação à mudança de estado. Isso ocorre, pois a taxa de amostragem é muito alta, em relação à quantidade de alterações de estado.

A Tabela 7 contém as informações da probabilidade de emissão por estado, ou seja, o que o algoritmo de aprendizado definiu como referência na definição do estado efetivo.

O Gráfico 2 demonstra uma comparação entre o estado efetivo com o estado definido pelo sistema de aprendizado. Nota-se a dificuldade do sistema em definir a diferença do estado de repouso com o estado de abastecimento.

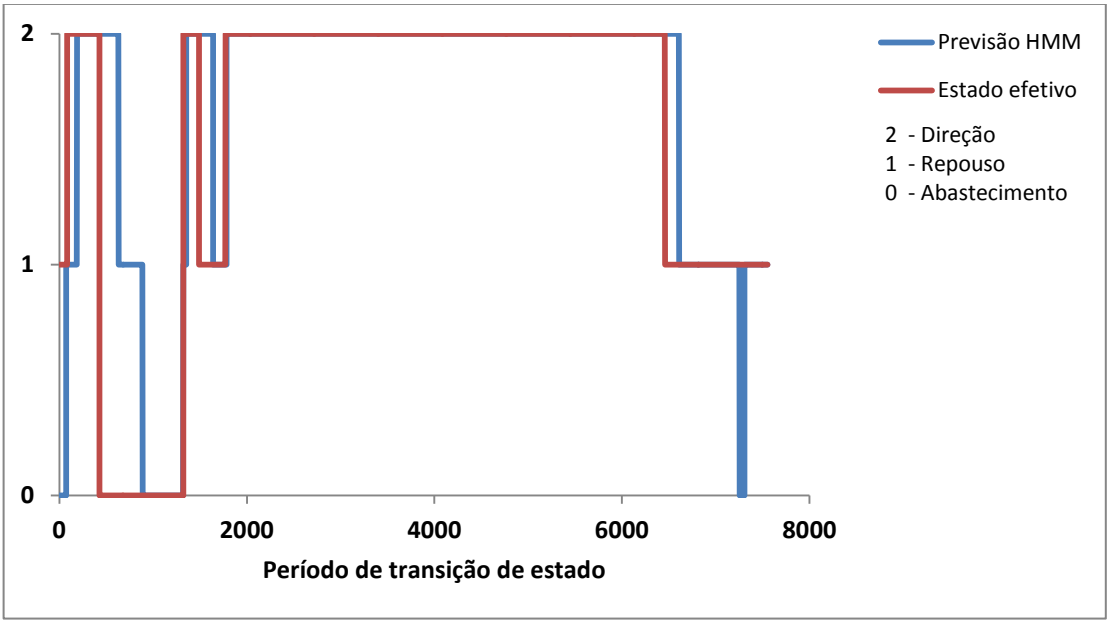


Gráfico 2 - Comparativo entre estado efetivo e estado definido pela primeira análise.

Fonte: O autor.

De forma a analisar a precisão do sistema, a Tabela 8 compara o estado efetivo com a estimativa. Ou seja, estando efetivamente em um estado, qual o estado definido pelo sistema.

	Repouso	Abastecimento	Dirigindo
Repouso	1033	125	301
Abastecimento	257	434	203
Dirigindo	149	0	5051

Tabela 8 - Tabela de definição do estado da primeira análise.

Fonte: O autor.

Este sistema obteve 70,8% de precisão na definição do estado de repouso, 48,5% para abastecimento e 97% para dirigindo. No total, a precisão do sistema foi de 86,3%. O tempo médio para transição de estado foi de 108 amostras.

4.2 SEGUNDA ANÁLISE

Para a segunda análise, utilizou-se uma taxa de amostragem de 1 amostra a cada 6 segundos. Para esta análise, foram utilizados os mesmo sinais da primeira, sendo eles: Acelerador, combustível, freio, porta, volume, chave de ignição, e movimento.

Após o treinamento e validação, obtiveram-se os seguintes resultados:

	Repouso	Abastecimento	Direção
Repouso	0,992	0,000	0,008
Abastecimento	0,003	0,992	0,006
Direção	0,003	0,008	0,989

Tabela 9 - Matriz de probabilidade de transição de estado da segunda análise.

Fonte: O autor.

	Aceleração	Combustível	Freio	Porta	Volume	Ignição	Movimento
Repouso	0,162	0,012	0,042	0,219	0,176	0,132	0,178
Abastecimento	0,085	0,072	0,06	0,223	0,585	0,077	0,097
Dirigindo	0,193	0,03	0,057	0,193	0,422	0,107	0,189

Tabela 10 - Matriz de probabilidade de emissão por estado da segunda análise.

Fonte: O autor.

Analisando a Tabela 9, identifica-se a redução da probabilidade de transição para o mesmo estado. Como analisado previamente, isso se dá devido ao fato de agora termos um sistema com menor taxa de amostragem.

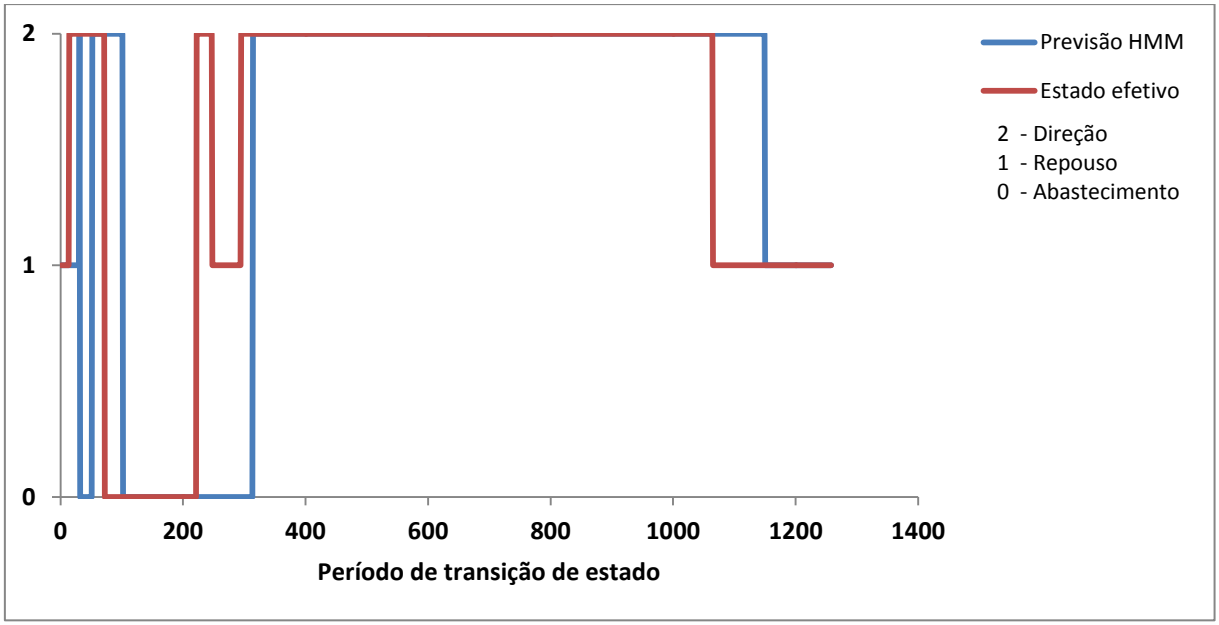


Gráfico 3 - Comparativo entre estado efetivo e estado definido pela segunda análise.

Fonte: O autor.

O Gráfico 3 faz a comparação entre o estado efetivo com o estado definido pelo sistema de aprendizado. Como o sistema possui uma menor taxa de amostragem, fica claro que o algoritmo de aprendizado tem dificuldade de definir corretamente estados de curta duração, além do maior período para transitar entre estados. Comparando a Tabela 11 com a Tabela 8, da análise de maior taxa de amostragem, nota-se a redução na precisão do sistema em relação à definição dos estados.

	Repouso	Abastecimento	Dirigindo
Repouso	479	898	82
Abastecimento	251	603	40
Dirigindo	808	53	4339

Tabela 11 - Tabela de definição do estado da segunda análise.

Fonte: O autor.

Este sistema obteve 32% de precisão para o sistema em repouso, 67,4% para abastecimento e 83,4% para dirigindo. Sendo 80,1% de acerto na definição de todos os estados. O tempo médio para transição de estado foi de 56 amostras.

4.3 TERCEIRA ANÁLISE

Para a terceira análise, utilizou-se uma taxa de amostragem de 1 amostra por segundo, com a remoção dos sinais identificados com alta correlação. Assim, os seguintes sinais foram verificados: Acelerador, combustível, freio, volume e chave de ignição.

Após o treinamento e validação, obteve-se os seguintes resultados:

	Repouso	Abastecimento	Direção
Repouso	0,999	0,001	0,000
Abastecimento	0,002	0,997	0,002
Direção	0,000	0,002	0,998

Tabela 12 - Matriz de probabilidade de transição de estado da terceira análise.

Fonte: O autor.

	Aceleração	Combustível	Freio	Volume	Ignição
Repouso	0,597	0,025	0,094	0,414	0,297
Abastecimento	0,050	0,320	0,009	0,701	0,397
Direção	0,517	0,008	0,212	0,706	0,000

Tabela 13 - Matriz de probabilidade de emissão por estado da terceira análise.

Fonte: O autor.

Novamente, devido ao fato do sistema ter uma maior taxa de amostragem, a Tabela 12 demonstra que a probabilidade do sistema manter-se no mesmo estado é muito maior que a de mudança de estado. Devido à remoção de dados do sistema, nota-se a alteração das probabilidades de emissão por estado ao comparar as tabelasTabela 13 eTabela 7.

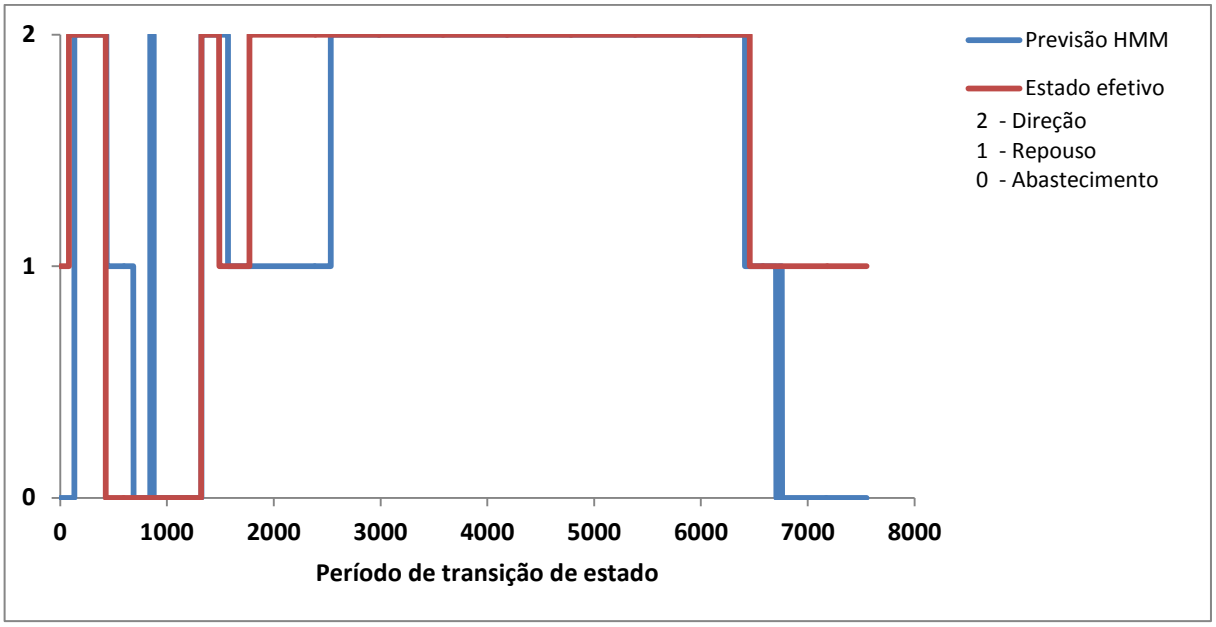


Gráfico 4 - Comparativo entre estado efetivo e estado definido pela terceira análise.

Fonte: O autor.

O Gráfico 4 - Comparativo entre estado efetivo e estado definido pela terceira análise. faz a comparação entre o estado efetivo e o estado definido pelo sistema de aprendizado. Verifica-se que este sistema transitou entre estados de maneira mais rápida na maioria dos casos, se comparado com a primeira análise.

	Repouso	Abastecimento	Dirigindo
Repouso	122	47	85
Abastecimento	0	120	30
Dirigindo	17	91	772

Tabela 14 - Tabela de definição do estado da terceira análise.

Fonte: O autor.

Obteve-se 48% de precisão no estado de repouso, 80% em abastecimento e 87,7% dirigindo. Sendo, no total, 71,8% de precisão total. O tempo médio para transição de estado foi de 277 amostras.

4.4 QUARTA ANÁLISE

Para a quarta análise, utilizou-se uma taxa de amostragem de 1 amostra a cada 6 segundos, sendo os sinais de estudo os seguintes: Acelerador, combustível, freio, volume e chave de ignição.

Após o treinamento e validação do sistema, obtiveram-se os seguintes resultados:

	Repouso	Abastecimento	Direção
Repouso	0,990	0,005	0,005
Abastecimento	0,000	0,994	0,006
Direção	0,002	0,002	0,995

Tabela 15 - Matriz de probabilidade de transição de estado da quarta análise.

Fonte: O autor.

	Aceleração	Combustível	Freio	Volume	Ignição
Repouso	0,117	0,046	0,061	0,647	0,026
Abastecimento	0,201	0,112	0,017	0,322	0,11
Direção	0,306	0,014	0,047	0,231	0,121

Tabela 16 - Matriz de probabilidade de emissão por estado da quarta análise.

Fonte: O autor.

A Tabela 15 demonstra a matriz de probabilidade de transição de estados. Nota-se a redução da probabilidade de transição para o mesmo estado se comparado com as análises de maior taxa de amostragem.

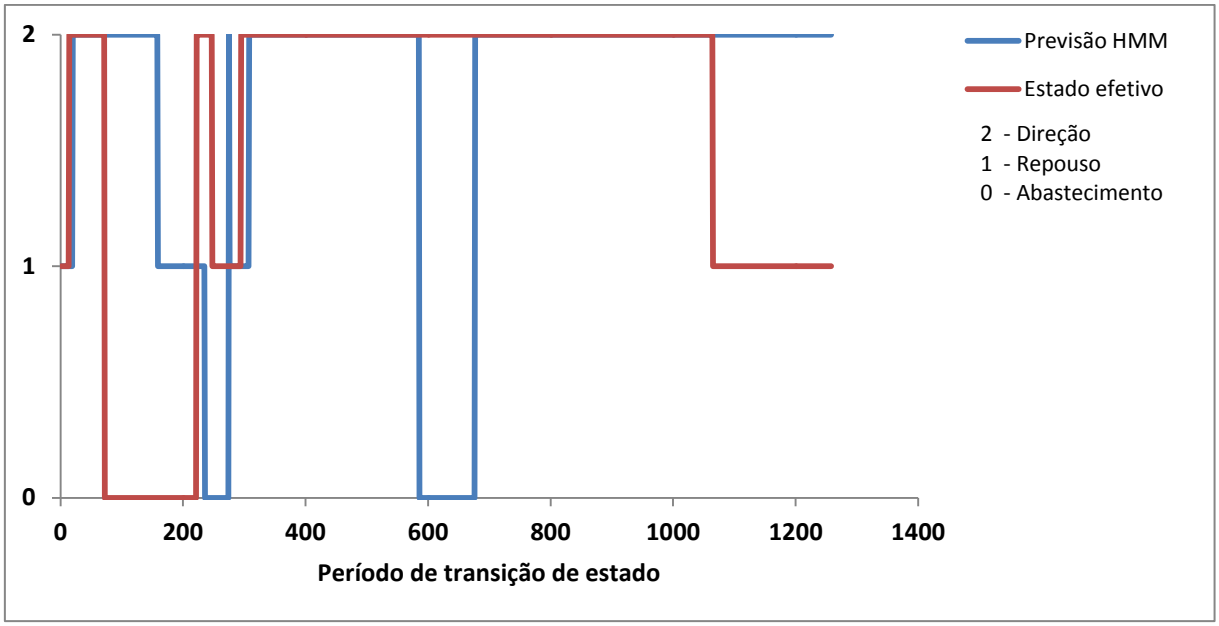


Gráfico 5 - Comparativo entre estado efetivo e estado definido pela quarta análise.

Fonte: O autor.

O Gráfico 5 demonstra um comparativo entre o estado efetivo e a previsão do HMM da quarta análise. Nota-se o maior tempo de transição entre estados, o que acaba reduzindo drasticamente a precisão do sistema. Além disso, com a redução da taxa de amostragem e de sinais, identifica-se que o sistema acabou definindo erroneamente o estado durante parte da viagem executada, sendo essa definição independente de troca recente de estado.

	Repouso	Abastecimento	Dirigindo
Repouso	32	27	195
Abastecimento	63	0	87
Dirigindo	33	103	718

Tabela 17 - Tabela de definição do estado da quarta análise.

Fonte: O autor.

Este sistema obteve a precisão de 12,6% de definição do estado de repouso, 0% para abastecimento e 84% para dirigindo. Sendo o total de 59,6% de precisão. O tempo médio para transição de estado foi de 57 amostras.

Através da análise dos resultados, identifica-se que a primeira análise, contendo maior taxa de amostragem e maior quantidade de sinais, demonstrou

maior precisão, na definição de estado comportamental do motorista no total da viagem. Em relação à terceira análise, ao remover dados de alta correlação, verifica-se que o sistema definiu a transição de estado de maneira mais rápida na maioria das transições, apesar de ter obtido a pior média no tempo de transição de estado. Nota-se, também, que os sistemas analisados com baixa taxa de amostragem obtiveram o melhor tempo médio de transição de estados, mas tiveram dificuldade em estimar estados com curta duração de tempo, reduzindo drasticamente sua precisão.

CONSIDERAÇÕES FINAIS

O comportamento de um motorista de caminhão em seu trabalho tem maior dinâmica do que apenas os estados que foram definidos no estudo. No entanto, identificou-se que, relativamente, com poucos sinais, é possível obter uma alta taxa de acerto sobre o seu comportamento.

Analisando os sinais, os estados do motorista em repouso e o reabastecimento do veículo têm pouca diferença entre si, caso o sinal de combustível não esteja presente ou demore a alterar. Devido a este fato, verifica-se que o sistema definiu o estado erroneamente em três das quatro análises antes da informação de combustível ser alterada. No entanto, ao iniciar a alteração dos sinais de combustível, a previsão foi realizada com sucesso, embora tenha havido certo atraso.

4.5 RECOMENDAÇÕES PARA TRABALHOS FUTUROS

De modo a aprofundar o estudo, pode-se utilizar uma maior quantidade de amostras tanto para treinamento quanto validação do sistema.

O estudo efetuado contempla apenas três dos mais diversos estados comportamentais de um motorista com seu caminhão. Portanto, outros estados como manutenção, carga e descarga, podem ser analisados.

Os mais diversos sinais e redes CAN podem ser adicionados ao estudo de modo a aprimorar o resultado, tanto na velocidade de identificação de troca de estado quanto na precisão do sistema.

Podem ser aplicados diversos métodos de aprendizado de máquina para a definição do seu estado comportamental, de modo a definir qual é o mais indicado para esta situação. Um bom candidato para comparação são as Redes Neurais Artificiais (RNA). Estas redes utilizam métodos de treinamento supervisionados, ao contrário da HMM aplicada neste trabalho. Os resultados obtidos podem ser utilizados como base para comparação da eficiência dos métodos competidores na identificação do estado provável do motorista.

REFERÊNCIAS

[1] Lei do motorista, Nº 13.103, DE 2 DE MARÇO DE 2015. Disponível em: http://www.planalto.gov.br/ccivil_03/_ato2015-2018/2015/lei/l13103.htm

[2] J. S. Espindola. Um Estudo sobre Modelos Ocultos de Markov HMM - Hidden Markov Model. Pontifícia Universidade Católica do Rio Grande do Sul. 2009

[3] L. R. Rabiner e B. H. Juang. An introduction to hidden markov models. IEEE ASSP Magazine, 3(1):4–16, 1986.

[4] H. T. Junior. Estudo dos protocolos de comunicação das arquiteturas eletroeletrônicas automotivas, com foco nas suas características e respectivas aplicações, visando o direcionamento para o uso adequado e customizado em cada categoria de veículo. Centro Universitário do Instituto Mauá de Tecnologia. 2010

[5] L. R. Rabiner. A tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. Proceedings of the IEEE. Fevereiro de 1989

[6] A. A. Guimarães. O Protocolo CAN: Entendendo e Implementando uma Rede de Comunicação Serial de Dados baseada no Barramento “Controller Area Network”. Disponível em alexag.com.br. Acessado em 14/08/18.

[6] Vector. Introduction to SAEJ1939. Disponível em: https://vector.com/portal/medien/cmc/application_notes/AN-ION-1-3100_Introduction_to_J1939.pdf. Acessado em 14/08/18.

[7] Portal Action. Cadeias de Markov. Disponível em: http://www.portalaction.com.br/processo-estocastico/cadeia-de-markov. Acessado em 23/09/18

[8] Modelo Oculto de Markov. https://pt.wikipedia.org/wiki/Modelo_oculto_de_Markov Acessado em 23/09/18

[9] Lussier E. F. Markov Models and Hidden Markov Models: A Brief Tutorial. International Computer Science Institute. Dezembro de 1998.

[10] CSS Electronics, CAN Bus explained – A simple intro - https://www.csselectronics.com/screen/page/simple-intro-to-can-bus/language/en. Acessado em 28/08/18.

[11] C. M. Grinstead; J. L. Snell. Introduction to Probability. American Mathematical Society. Acessado em 02/10/2018.

[12] Blunsom P. Hidden Markov Models. 2004.

[13] http://www.ti.com/lit/an/sloa101b/sloa101b.pdf

[14] Visser, I. e Speekenbrink, M. depmixS4: An R Package for Hidden Markov Models. Agosto 2010

[15] Moving Averages stockcharts.com/school/doku.php?id=chart_school:technical_indicators:moving_averages. Acessado em 08/10/2018

[16] Vakati K. Driver Telematics Analysis. San Jose State University. 2015.

[17] C. Hwang, M. Chen, C. Shih, H. Chen and W. K. Liu, "Apply Scikit-Learn in Python to Analyze Driver Behavior Based on OBD Data". IEEE. Acessado em 12/11/18

[18] The R project for statistical computing. https://www.r-project.org/ Acessado em 28/08/2018

5 ANEXO

Códigos em C para tratamento dos sinais:

Filtro Acelerador:

```
#include <stdio.h>
#include <stdlib.h>

void main()
{
   FILE *a;
   char str[1000000];
   float t;
   int i;
   char id[100];
   char op[100];
   char da[100];
   int oito;
   int d[8];
   char outros[9][100];
   int interessa;

   if ((a=fopen("./sinais/Acelerador.asc","r")) == NULL) {
       printf("\nErro lendo Leitura1.asc");
       exit (0);
   }

   while (!feof(a))
   {
       fgets(str, sizeof(str), a);

       //printf("%s",str);

       sscanf(str,"%f %i %s %s %s %d %x %x %x %x %x %x %x %x  %s %s %s  %s %s %s  %s %s
%s ",&t, &i, id, op, da, &oito, &d[0],&d[1],&d[2],&d[3],&d[4],&d[5],
&d[6],&d[7],outros[0],outros[1],outros[2],outros[3],outros[4],outros[5],outros[6],outros[7],outros[8]);

      printf("%d\n", interessa);

      interessa=0;

      interessa=d[7];

   }

}
```

Média Móvel Exponencial Acelerador:

```
#include <stdio.h>
#include <stdlib.h>

void main()
{
   FILE *a;
   char str[1000000];
```

```
  float i;
  int j=0;
  float mme[1000000];
  float buff=0;

  if ((a=fopen("./filtro/Acelerador_filtrado.asc","r")) == NULL) {
     printf("\nErro lendo Leitura1.asc");
     exit (0);
  }
  while (!feof(a)){
      fgets(str, sizeof(str), a);
          sscanf(str, "%e ", &i);
      mme[j] = (i-buff)*0.18 + buff;
  buff=mme[j];
  printf("%f\n", mme[j]);
 j=j+1;
  }

}
```

Discretização Acelerador:

```
#include <stdio.h>
#include <stdlib.h>

void main()
{
  FILE *a;
  char str[1000000];
  float i;
  int j=0;
  int b=0;
  int contador=0;
  int dis[1000000];

  if ((a=fopen("./mme/Med_exp_acelerador.asc","r")) == NULL) {
     printf("\nErro lendo Leitura1.asc");
     exit (0);
  }
  while (!feof(a)){
      fgets(str, sizeof(str), a);
          sscanf(str, "%f ", &i);
          if(i== 229.5){
          dis[j]= 0;
          }
          if(i== 0){
      dis[j]= 0;
      }
          if(i>0 && i<229.5f){
      dis[j]= 2;
      }
  if(contador == 100){
  printf("%d\n", dis[j]);
  contador = 0;
}
 contador = contador + 1;
 j=j+1;
  }
}
```

Filtro Combustível:

```
#include <stdio.h>
#include <stdlib.h>

void main()
{
   FILE *a;
   char str[10000];
   float t;
   int i;
   char id[100];
   char op[100];
   char da[100];
   int oito;
   int d[8];
   char outros[9][100];
   int interessa;

   if ((a=fopen("./sinais/Combustivel.asc","r")) == NULL) {
       printf("\nErro lendo Leitura1.asc");
       exit (0);
   }

   while (!feof(a))
   {
       fgets(str, sizeof(str), a);

       //printf("%s",str);

       sscanf(str,"%f %i %s %s %s %d %x %x %x %x %x %x %x %x  %s %s %s  %s %s %s  %s %s
%s ",&t, &i, id, op, da, &oito, &d[0],&d[1],&d[2],&d[3],&d[4],&d[5],
&d[6],&d[7],outros[0],outros[1],outros[2],outros[3],outros[4],outros[5],outros[6],outros[7],outros[8]);

      printf("%d\n", interessa);

      interessa=0;

      interessa=d[0];

   }

}
```

Média Móvel Exponencial Combustível:

```
#include <stdio.h>
#include <stdlib.h>

void main()
{
   FILE *a;
   char str[1000000];
   int i;
   int j=0;
   float mme[1000000];

   if ((a=fopen("./filtro/Combustivel_filtrado.asc","r")) == NULL) {
       printf("\nErro lendo Leitura1.asc");
       exit (0);
```

```
  }

  while (!feof(a))
  {
     fgets(str, sizeof(str), a);

          sscanf(str, "%i ", &i); //converte para um valor numérico
      mme[j] = i;
  printf("%f\n", mme[j]);
 j=j+1;

}
}
```

Discretização Combustível:

```
#include <stdio.h>
#include <stdlib.h>

void main()
{
  FILE *a;
  char str[1000000];
  float i;
  int j=0;
  float buff=0;
  int contador=0;
  int dis[1000000];

  if ((a=fopen("./mme/Med_exp_combustivel.asc","r")) == NULL) {
     printf("\nErro lendo Leitura1.asc");
     exit (0);
  }
  while (!feof(a)){
      fgets(str, sizeof(str), a);
          sscanf(str, "%f ", &i);
          if(i == buff){
          dis[j]= 0;
          }
          if(i != buff){
      dis[j]= 2;
          }
  if(contador == 1){
  printf("%d\n", dis[j]);
  contador = 0;
  }
 j=j+1;
 contador=contador+1;
 buff=i;
  }
}
```

Filtro Freio:

```
#include <stdio.h>
#include <stdlib.h>

void main()
{
  FILE *a;
```

```
   char str[10000];
   float t;
   int i;
   char id[100];
   char op[100];
   char da[100];
   int oito;
   int d[8];
   char outros[9][100];
   int interessa;

   if ((a=fopen("./sinais/Freio.asc","r")) == NULL) {
       printf("\nErro lendo Leitura1.asc");
       exit (0);
   }

   while (!feof(a))
   {
       fgets(str, sizeof(str), a);

       //printf("%s",str);

       sscanf(str,"%f %i %s %s %s %d %x %x %x %x %x %x %x %x  %s %s %s  %s %s %s  %s %s
%s ",&t, &i, id, op, da, &oito, &d[0],&d[1],&d[2],&d[3],&d[4],&d[5],
&d[6],&d[7],outros[0],outros[1],outros[2],outros[3],outros[4],outros[5],outros[6],outros[7],outros[8]);

       printf("%d\n", interessa);

       interessa=0;

       interessa=d[6];
       }

}
```

Média Móvel Exponencial Freio:

```
#include <stdio.h>
#include <stdlib.h>

void main()
{
   FILE *a;
   char str[1000000];
   float i;
   float buff=0;
   int j=0;
   float mme[1000000];

   if ((a=fopen("./filtro/Freio_filtrado.asc","r")) == NULL) {
       printf("\nErro lendo Leitura1.asc");
       exit (0);
   }

   while (!feof(a))
   {
       fgets(str, sizeof(str), a);
            sscanf(str, "%e ", &i); //converte para um valor numérico
        mme[j] = (i-buff)*0.18 + buff;
   buff=mme[j];
```

```
   printf("%f\n", mme[j]);
  j=j+1;
   }
}
```

Discretização Freio:

```
#include <stdio.h>
#include <stdlib.h>

void main()
{
   FILE *a;
   char str[1000000];
   float i;
   int j=0;
   int contador=0;
   int dis[1000000];

   if ((a=fopen("./mme/Med_exp_freio.asc","r")) == NULL) {
      printf("\nErro lendo Leitura1.asc");
      exit (0);
   }
   while (!feof(a)){
        fgets(str, sizeof(str), a);
            sscanf(str, "%f ", &i);
            if(i== 229.5){
            dis[j]= 0;
            }
            if(i== 0){
        dis[j]= 0;
        }
            if(i>0 && i<229.5f){
        dis[j]= 2;
        }
   if(contador == 100){
   printf("%d\n", dis[j]);
   contador = 0;
}
  contador = contador + 1;
  j=j+1;
   }
}
```

Filtro Porta:

```
#include <stdio.h>
#include <stdlib.h>

void main()
{
   FILE *a;
   char str[10000];
   float t;
   int i;
   char id[100];
   char op[100];
   char da[100];
   int oito;
```

```
  int d[8];
  char outros[9][100];
  int interessa;

  if ((a=fopen("./sinais/Porta.asc","r")) == NULL) {
      printf("\nErro lendo Leitura1.asc");
      exit (0);
  }

  while (!feof(a))
  {
      fgets(str, sizeof(str), a);

      //printf("%s",str);

      sscanf(str,"%f %i %s %s %s %d %x %x %x %x %x %x %x %x  %s %s %s  %s %s %s  %s %s
%s ",&t, &i, id, op, da, &oito, &d[0],&d[1],&d[2],&d[3],&d[4],&d[5],
&d[6],&d[7],outros[0],outros[1],outros[2],outros[3],outros[4],outros[5],outros[6],outros[7],outros[8]);

      printf("%d\n", interessa);

      interessa=0;

      interessa=d[5];

  }
}
```

Média Móvel Exponencial Porta:

```
#include <stdio.h>
#include <stdlib.h>

void main()
{
  FILE *a;
  char str[1000000];
  float i;
  float buff = 0;
  int j=0;
  float mme[1000000];

  if ((a=fopen("./filtro/Porta_filtrado.asc","r")) == NULL) {
      printf("\nErro lendo Leitura1.asc");
      exit (0);
  }

  while (!feof(a))
  {
      fgets(str, sizeof(str), a);
          sscanf(str, "%e ", &i); //converte para um valor numérico
       mme[j] = (i-buff)*0.9 + buff;
  buff=mme[j];
  printf("%f\n", mme[j]);
 j=j+1;
  }
 }
```

Discretização Porta:

```
#include <stdio.h>
#include <stdlib.h>

void main()
{
   FILE *a;
   char str[1000000];
   float i;
   int j=0;
   int contador =0;
   int dis[1000000];

   if ((a=fopen("./mme/Med_exp_porta.asc","r")) == NULL) {
      printf("\nErro lendo Leitura1.asc");
      exit (0);
   }
   while (!feof(a)){
       fgets(str, sizeof(str), a);
           sscanf(str, "%f ", &i);
           if(i < 140){ // abertas
           dis[j]= 0;
           }
           if(i>140 && i<145){ // uma aberta
       dis[j]= 1;
       }
           if(i > 145){ // trancadas
       dis[j]= 2;
       }
   if(contador == 2){
   printf("%d\n", dis[j]);
   contador = 0;
}
  contador = contador + 1;
  j=j+1;
   }
}
```

Filtro Volume:

```
#include <stdio.h>
#include <stdlib.h>

void main()
{
   FILE *a;
   char str[10000];
   float t;
   int i;
   char id[100];
   char op[100];
   char da[100];
   int oito;
   int d[8];
   char outros[9][100];
   int interessa;

   if ((a=fopen("./sinais/Radio_volume.asc","r")) == NULL) {
      printf("\nErro lendo Leitura1.asc");
      exit (0);
```

```
  }

  while (!feof(a))
  {
     fgets(str, sizeof(str), a);

     //printf("%s",str);

     sscanf(str,"%f %i %s %s %s %d %x %x %x %x %x %x %x %x  %s %s %s  %s %s %s  %s %s
%s ",&t, &i, id, op, da, &oito, &d[0],&d[1],&d[2],&d[3],&d[4],&d[5],
&d[6],&d[7],outros[0],outros[1],outros[2],outros[3],outros[4],outros[5],outros[6],outros[7],outros[8]);

     printf("%d\n", interessa);

     interessa=0;

     interessa=d[2];

  }
}
```

Média Móvel Exponencial Volume:

```
#include <stdio.h>
#include <stdlib.h>

void main()
{
  FILE *a;
  char str[1000000];
  float i;
  float buff =0;
  int j=0;
  float mme[1000000];

  if ((a=fopen("./filtro/Radio_volume_filtrado.asc","r")) == NULL) {
     printf("\nErro lendo Leitura1.asc");
     exit (0);
  }

  while (!feof(a))
  {
     fgets(str, sizeof(str), a);

          sscanf(str, "%e ", &i); //converte para um valor numérico
          mme[j] = (i-buff)*0.7 + buff;
  buff=mme[j];
  printf("%f\n", mme[j]);
 j=j+1;
  }
}
```

Discretização Volume:

```
#include <stdio.h>
#include <stdlib.h>

void main()
{
```

```
   FILE *a;
   char str[1000000];
   float i;
   int j=0;
   int contador=0;
   int dis[1000000];

   if ((a=fopen("./mme/Med_exp_radio_volume.asc","r")) == NULL) {
      printf("\nErro lendo Leitura1.asc");
      exit (0);
   }
   while (!feof(a)){
       fgets(str, sizeof(str), a);
           sscanf(str, "%f ", &i);
           if(i < 30){
           dis[j]= 0;
           }
           if(i > 30 && i < 160){
       dis[j]= 1;
       }
           if(i > 160){
       dis[j]= 2;
       }
   if(contador == 5){
   printf("%d\n", dis[j]);
   contador = 0;
}
  contador = contador + 1;
  j=j+1;
   }
}
```

Filtro Ignição:

```
#include <stdio.h>
#include <stdlib.h>

void main()
{
   FILE *a;
   char str[10000];
   float t;
   int i;
   char id[100];
   char op[100];
   char da[100];
   int oito;
   int d[8];
   char outros[9][100];
   int interessa;

   if ((a=fopen("./sinais/Vehicle_mode.asc","r")) == NULL) {
      printf("\nErro lendo Leitura1.asc");
      exit (0);
   }

   while (!feof(a))
   {
      fgets(str, sizeof(str), a);
```

```
      //printf("%s",str);

      sscanf(str,"%f %i %s %s %s %d %x %x %x %x %x %x %x %x  %s %s %s  %s %s %s  %s %s
%s ",&t, &i, id, op, da, &oito, &d[0],&d[1],&d[2],&d[3],&d[4],&d[5],
&d[6],&d[7],outros[0],outros[1],outros[2],outros[3],outros[4],outros[5],outros[6],outros[7],outros[8]);

     printf("%d\n", interessa);

     interessa=0;

     interessa=d[3];

     }

}
```

Média Móvel Exponencial Ignição:

```
#include <stdio.h>
#include <stdlib.h>

void main()
{
  FILE *a;
  char str[1000000];
  float i;
  float buff =0;
  int j=0;
  float mme[1000000];

  if ((a=fopen("./filtro/Vehicle_mode_filtrado.asc","r")) == NULL) {
     printf("\nErro lendo Leitura1.asc");
     exit (0);
  }

  while (!feof(a))
  {
     fgets(str, sizeof(str), a);

          sscanf(str, "%e ", &i); //converte para um valor numérico
      mme[j] = (i-buff)*0.18 + buff;
  buff=mme[j];
  printf("%f\n", mme[j]);
 j=j+1;
  }
}
```

Discretização Ignição:

```
#include <stdio.h>
#include <stdlib.h>

void main()
{
  FILE *a;
  char str[1000000];
  float i;
  int j=0;
  int contador=0;
```

```
   int dis[1000000];

   if ((a=fopen("./mme/Med_exp_vehicle_mode.asc","r")) == NULL) {
      printf("\nErro lendo Leitura1.asc");
      exit (0);
   }
   while (!feof(a)){
       fgets(str, sizeof(str), a);
           sscanf(str, "%f ", &i);
           if(i<=42){
           dis[j]= 0; // desligado
           }
           if(i>42 && i<65){
       dis[j]= 1; //acessorios
       }
           if(i>65){
       dis[j]= 2; // ligado
       }
   if(contador == 100){
   printf("%d\n", dis[j]);
   contador = 0;
}
  contador = contador + 1;
  j=j+1;
   }
}
```

Filtro Velocidade:

```
#include <stdio.h>
#include <stdlib.h>

void main()
{
   FILE *a;
   char str[10000];
   float t;
   int i;
   char id[100];
   char op[100];
   char da[100];
   int oito;
   int d[8];
   char outros[9][100];
   int interessa;

   if ((a=fopen("./sinais/Velocidade.asc","r")) == NULL) {
      printf("\nErro lendo Leitura1.asc");
      exit (0);
   }

   while (!feof(a))
   {
      fgets(str, sizeof(str), a);

      //printf("%s",str);

      sscanf(str,"%f %i %s %s %s %d %x %x %x %x %x %x %x %x  %s %s %s  %s %s %s  %s %s
%s ",&t, &i, id, op, da, &oito, &d[0],&d[1],&d[2],&d[3],&d[4],&d[5],
&d[6],&d[7],outros[0],outros[1],outros[2],outros[3],outros[4],outros[5],outros[6],outros[7],outros[8]);
```

```
    printf("%d\n", interessa);

    interessa=0;

    interessa=d[2]<<8 | d[3];
    }
}
```

Média Móvel Exponencial Velocidade:

```
#include <stdio.h>
#include <stdlib.h>

void main()
{
  FILE *a;
  char str[1000000];
  float i;
  float buff = 0;
  int j=0;
  float mme[1000000];

  if ((a=fopen("./filtro/Med_velocidade_filtrado.asc","r")) == NULL) {
     printf("\nErro lendo Leitura1.asc");
     exit (0);
  }

  while (!feof(a))
  {
     fgets(str, sizeof(str), a);

          sscanf(str, "%e ", &i); //converte para um valor numérico
      mme[j] = (i-buff)*0.5 + buff;
  buff=mme[j];
  printf("%f\n", mme[j]);
 j=j+1;
  }
 }
```

Discretização Velocidade:

```
#include <stdio.h>
#include <stdlib.h>

void main()
{
  FILE *a;
  char str[1000000];
  float i;
  int j=0;
  int contador=0;
  int dis[1000000];

  if ((a=fopen("./filtro/Velocidade_filtrado.asc","r")) == NULL) {
     printf("\nErro lendo Leitura1.asc");
     exit (0);
  }
  while (!feof(a)){
```

```
        fgets(str, sizeof(str), a);
            sscanf(str, "%f ", &i);
            if(i== 65535){
            dis[j]= 0;
            }
            if(i== 0){
        dis[j]= 0;
        }
            if(i>0 && i<65535){
        dis[j]= 2;
        }
   if(contador == 33){
   printf("%d\n", dis[j]);
   contador = 0;
}
  contador = contador + 1;
  j=j+1;
   }
}
```

Dendrograma em R:

```
D <- read.table("dadosm1s.csv", sep=";",header=TRUE)
d<-D[,-9]
d.use<-d[,-1]

d.dist=dist(d.use)
d.hclust=hclust(d.dist)
plot(d.hclust)
```

Treinamento de HMM em R:

```
library(depmixS4)
library(quantmod)
library(qcc)
D <- read.table("dadosm1s.csv", sep=";",header=TRUE)
a<-D$Acelerador
c<-D$Combustivel
f<-D$Freio
p<-D$Porta
v<-D$Volume
i<-D$Ignicao
m<-D$Movimento
a<-ewmaSmooth(x=D$Time, y=D$Acelerador, lambda=0.01) # Aplicação da MME
c<-ewmaSmooth(x=D$Time, y=D$Combustivel, lambda=0.01)
f<-ewmaSmooth(x=D$Time, y=D$Freio, lambda=0.01)
p<-ewmaSmooth(x=D$Time, y=D$Porta, lambda=0.01)
v<-ewmaSmooth(x=D$Time, y=D$Volume, lambda=0.01)
i<-ewmaSmooth(x=D$Time, y=D$Ignicao, lambda=0.01)
m<-ewmaSmooth(x=D$Time, y=D$Movimento, lambda=0.01)
a<-a$y
c<-c$y
f<-f$y
p<-p$y
v<-v$y
i<-i$y
m<-m$y
```

```
HMM<-depmix(list(a~1,c~1,f~1, p~1, v~1, i~1,
m~1),nstates=3,family=list(gaussian(),gaussian(),gaussian(), gaussian(), gaussian(), gaussian(),
gaussian()),ntimes=6396) # Aplicação da função depmix
HMMfit<-fit(HMM, verbose = FALSE)
HMMpost<-posterior(HMMfit)
par(mfrow=c(3,3))
plot (a, col=2)
plot(c, col=3)
plot(f, col=4)
plot(p, col=5)
plot(v, col=6)
plot(i, col=7)
plot(m, col=8)
plot(HMMpost$state) # estado mais provável de acordo com o HMM
plot(D$Estado) # estado "real" para comparação da eficiência do método
print(HMMfit)
```

Validação do sistema em R:

```
F <- read.table("dadosm1sfinal.csv", sep=";",header=TRUE)
a1<-F$Acelerador
c1<-F$Combustivel
f1<-F$Freio
p1<-F$Porta
v1<-F$Volume
i1<-F$Ignicao
m1<-F$Movimento
a1<-ewmaSmooth(x=F$Time, y=F$Acelerador, lambda=0.01)
c1<-ewmaSmooth(x=F$Time, y=F$Combustivel, lambda=0.01)
f1<-ewmaSmooth(x=F$Time, y=F$Freio, lambda=0.01)
p1<-ewmaSmooth(x=F$Time, y=F$Porta, lambda=0.01)
v1<-ewmaSmooth(x=F$Time, y=F$Volume, lambda=0.01)
i1<-ewmaSmooth(x=F$Time, y=F$Ignicao, lambda=0.01)
m1<-ewmaSmooth(x=F$Time, y=F$Movimento, lambda=0.01)
a1<-a1$y
c1<-c1$y
f1<-f1$y
p1<-p1$y
v1<-v1$y
i1<-i1$y
m1<-m1$y
HMMfinal<-depmix(list(a1~1,c1~1,f1~1, p1~1, v1~1, i1~1,
m1~1),nstates=3,family=list(gaussian(),gaussian(),gaussian(), gaussian(), gaussian(), gaussian(),
gaussian()),ntimes=7553)
HMMfinal <- setpars(HMMfinal,getpars(HMMfit)) #aplicação dos parâmetros de aprendizado
viterbi(HMMfinal) #Gerando uma tabela de valores
```