SENSORES VIRTUAIS PARA A INDÚSTRIA AUTOMOBILÍSTICA BASEADOS EM REDES NEURAIS ARTIFICIAIS

Iron Tessaro
Universidade Federal do Paraná (PPGEE)
Av. Coronel Francisco H. dos Santos 100, Jardim das Américas – Curitiba-PR
irontessaro@gmail.com

Leandro S. Coelho
Universidade Federal do Paraná (PPGEE)
Av. Coronel Francisco H. dos Santos 100, Jardim das Américas – Curitiba-PR
leandro.coelho@pucpr.br

Helon V. H. Ayala
Pontifícia Universidade Católica do Rio de Janeiro (DEM)
R. Marquês de São Vicente 225, Gávea - Rio de Janeiro-RJ
helon@puc-rio.br

RESUMO

Um dos problemas existentes na indústria automobilística é a demora entre ser detectada a necessidade de novos sensores nos automóveis e a real produção dos mesmos. Esse problema se deve a vários fatores como tempo de projeto, custo de produção e mesmo ao espaço físico para a inserção de novos componentes em produtos já em fabricação, o que muitas vezes inviabiliza permanentemente a iniciativa ou torna o custo elevado quando não há alternativa a adição desse novo componente. Para resolver esse problema, uma solução em *software* foi apresentada e testada, aplicando-se redes neurais artificiais, uma abordagem da Computação Natural da subárea denominada Inteligência Artificial, emulando os sensores desejados e os comparando com dados de sensores instrumentados em laboratórios de testes. É demonstrado que esses sensores virtuais podem se utilizar de diferentes grandezas físicas, medidas por sensores reais, para emular outras grandezas, reduzindo custo, tempo e complexidade na adição de novos componentes.

PALAVRAS CHAVE. Sensores Virtuais, Redes Neurais Artificiais, Indústria Automotiva.

Tópicos. IND – PO na Indústria.

ABSTRACT

One of the problems in automotive industry is the delay between detecting the need for new automobile sensors and their manufacturing. This problem is due to several factors like project lead time, production cost and even the physical space for new component insertion in products already in production, which sometimes turns the initiative permanently unfeasible or expensive when there is no alternative than adding this new component. In order to solve this problem, a software solution has been presented by applying artificial neural networks, which is a Natural Computation approach within the sub-area called Artificial Intelligence, to emulate the desired sensors and comparing them to instrumented sensor data in test laboratories. It is demonstrated that those virtual sensors can use different physical values from real sensors in order to emulate other values, reducing cost, delay and complexity to the addition of new components.

KEYWORDS. Virtual Sensors, Artificial Neural Networks, Automotive Industry.

Paper topics. IND – OR in Industry.

https://proceedings.science/p/85563?lang=pt-br

1. Introdução

O aumento da competitividade global e a convergência de desempenho e especificação dos produtos são gatilhos importantes para a inovação contínua, visando manter a competitividade em longo prazo. Isso se aplica especialmente a indústrias de alta tecnologia, tais como as indústrias automobilísticas, onde o sucesso depende fortemente da inovação contínua. Práticas de desenvolvimento de produto eficientes tornam-se necessárias para combater o histórico de investimentos extraordinariamente altos no desenvolvimento de novos produtos [Clark et al. 1987]. Dentre os grupos de produtos desenvolvidos pela indústria automotiva, um dos mais notáveis é o dos sensores. Atualmente, cada veículo possui, em média, 80 sensores embarcados. Devido a tecnologias como a internet das coisas, é esperado que o número de sensores nos carros seja maior que 200. Esse número se traduz para, aproximadamente, 22 bilhões de sensores usados na indústria automotiva por ano, para 2020 [Pinelis 2013].

Com a explosão no campo de Big Data e a introdução de normas de emissão mais rigorosas em média a cada quatro anos, é necessário que as companhias não apenas melhorem o consumo de combustível, mas também forneça valor agregado aos seus clientes como desempenho do motor e relatório de dados em intervalos regulares. Uma solução para os dois problemas é por meio da instalação de uma variedade de sensores no motor, mas inserir uma vasta quantidade deles não é plausível devido ao custo. Sensores virtuais, por outro lado, podem substituir sensores físicos reduzindo o custo enquanto capturam dados essenciais do motor [Rastogi e Saini 2017]. Eles podem ser usados em aplicações como assistentes de direção [Hirsenkorn et al. 2016] e em unidades de controle eletrônicas de motores, substituindo sensores físicos por modelos em *software* com baixo custo por motor [Morey 2015]. Com a rápida mudança dos requisitos e aumento da necessidade de novos sensores, para diferentes aplicações, torna-se inviável a produção e readequação dos projetos para cada um deles, quando inseridos na forma de *hardware*. Uma maneira mais barata e rápida para inseri-los nos projetos é a emulação de sensores virtuais via *software*, utilizando sensores já existentes para estimar outras grandezas físicas relacionadas [Zaharia e Clenci 2013]. Outro aspecto importante de sensores virtuais está na sua manutenção e na sua confiabilidade: como são virtuais, esses sensores não demandam manutenção, e seus comportamentos só se alteram se os sensores reais, que fazem as vezes de entrada do sensor modelado, alterarem [Pathak e Halale 2017].

A contribuição desse artigo é mostrar a viabilidade da aplicação de sensores virtuais, na indústria automobilística, utilizando redes neurais. São utilizados dados reais de um motor diesel, coletados em laboratório, para a comparação dos resultados e treinamento da rede.

O restante do artigo encontra-se estruturado em cinco seções. Na seção 2, os fundamentos das redes neurais multicamada e do método de treinamento são apresentados. A seguir, na seção 3 são detalhados os resultados da aplicação das redes neurais em um estudo de caso da indústria automotiva. Após, a conclusão e pesquisa futura são mencionados na seção 4.

2. Fundamentos das redes neurais multicamada

Segundo [Nielsen 2015] uma rede neural é formada pela conexão de dois ou mais elementos matemáticos chamados neurônios artificiais. Esses neurônios funcionam como funções que recebem diversas entradas e produzem uma única saída, como apresentado na Figura 1.

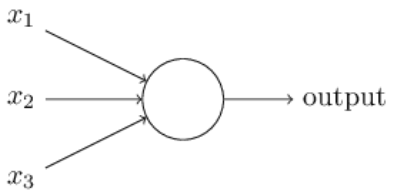


Figura 1 - Modelo geral de um neurônio artificial
Fonte: Adaptado de [Nielsen 2015]

Para cada entrada x_i do neurônio artificial, um peso w_i é atribuído, assim como uma única constante b para o neurônio. As entradas ponderadas e a constante são somadas, resultando em z, como apresentado pela equação (1).

$$z = \sum_i w_i x_i + b \tag{1}$$

Esse somatório ponderado de entradas com a constante é válido para todos os modelos de neurônios artificiais. O que difere um tipo de neurônio de outro é a chamada função de ativação. A função de ativação é responsável por inserir a não-linearidade na equação (1), fazendo com que uma rede neural possa representar qualquer tipo de função [Nielsen 2015]. Existem diversos tipos de funções de ativação, sendo a função sigmóide, apresentada pela equação (2), a mais comum entre elas.

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{2}$$

Inserindo a equação (1) na equação (2), obtém-se a saída do neurônio artificial, com função de ativação sigmóide, por meio da equação (3).

$$a = \frac{1}{1 + e^{-(\sum_i w_i x_i + b)}} \tag{3}$$

A representação de uma função $f(x_1, x_2, \ldots, x_n)$ por um neurônio artificial é dada ajustando-se os pesos de forma que, para um dado conjunto de entradas $\{x_1, x_2, \ldots, x_n\}$, a seja aproximadamente igual a $f(x_1, x_2, \ldots, x_n)$ [Nielsen 2015].

Um neurônio somente não é suficiente para representar qualquer tipo de função. É necessário agrupar vários deles em forma de camadas, e conectar camadas entre si, constituindo uma rede neural multicamada, o que aumenta o número de variáveis (pesos) a serem ajustados e a complexidade das funções intermediárias dentro da própria rede [Nielsen 2015]. Considerando o modelo do neurônio artificial da Figura 1, apresenta-se uma rede neural multicamada na Figura 2.

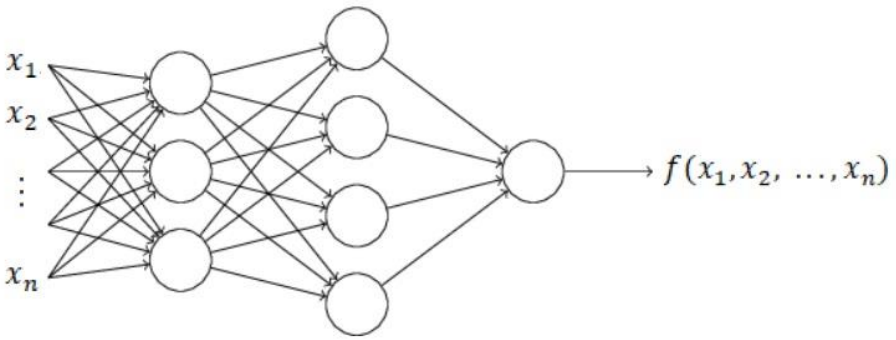


Figura 2 - Modelo geral de uma rede neural multicamada
Fonte: Adaptado de [Nielsen 2015]

Não existe uma regra geral de quantas camadas e quantos neurônios em cada camada são necessários para obter um melhor resultado, já que isso depende dos dados disponíveis [Nielsen 2015]. Um método normalmente utilizado é o de construir a primeira camada com o número de neurônios igual ao número de variáveis de entrada do modelo e a saída, obrigatoriamente, com o número de neurônios igual ao número de variáveis de saída. A partir desse ponto, trabalha-se empiricamente com as chamadas camadas ocultas (isto é, camadas entre as camadas de entrada e saída), sempre procurando o menor número possível de elementos [Goodfellow et al. 2016].

O termo “aprender” para uma rede neural significa ajustar os pesos e constantes de todos os neurônios para que ela consiga representar uma função que relaciona dados de entrada com dados de saída. A técnica mais utilizada para realizar o ajuste dos pesos e constantes é o algoritmo de retropropagação do erro juntamente com algum método de otimização [Nielsen 2015].

2.1. Descida de encosta

Segundo [Ruder 2017] a descida de encosta é um dos mais populares algoritmos de otimização e de longe a forma mais comum de otimizar um rede neural. Esse algoritmo se baseia no gradiente do espaço de busca para encontrar um mínimo local, incrementando passos proporcionalmente ao gradiente. É necessário definir uma função a ser otimizada antes de aplicar o algoritmo. No caso de redes neurais, o mais comum é utilizar o MSE (*mean squared error*), definido pela equação (4). Nessa equação, $\boldsymbol{w}$ é o conjunto de todos os pesos da rede, $\boldsymbol{b}$ é o conjunto de todas as constantes dos neurônios, n é o número total de amostras de entrada, $\boldsymbol{a}$ é o vetor de saídas da rede neural quando $\boldsymbol{x}$ é a entrada e $f(\boldsymbol{x})$ é o vetor de saídas esperadas [Nielsen 2015], tal que

$$C(\boldsymbol{w},\boldsymbol{b}) = \frac{1}{2n}\sum_{\boldsymbol{x}} \|f(\boldsymbol{x}) - \boldsymbol{a}(\boldsymbol{w},\boldsymbol{b},\boldsymbol{x})\|^2 \tag{4}$$

O valor esperado do erro médio quadrático, ao final do treinamento da rede neural, é nulo, pois isso significa que $f(\boldsymbol{x})$ e $\boldsymbol{a}(\boldsymbol{w},\boldsymbol{b},\boldsymbol{x})$ são iguais (isto é, a saída da rede é igual à saída esperada). Supondo $C(\boldsymbol{w},\boldsymbol{b})$ como a função a ser otimizada pelo algoritmo descida de encosta, define-se a variação da função $C(\boldsymbol{w},\boldsymbol{b})$em relação a uma variação em $\boldsymbol{w}$ e $\boldsymbol{b}$ de acordo com a equação (5) [Nielsen 2015], tal que

$$\Delta C(\boldsymbol{w},\boldsymbol{b}) \approx \frac{\partial C}{\partial \boldsymbol{w}}\Delta\boldsymbol{w} + \frac{\partial C}{\partial \boldsymbol{b}}\Delta\boldsymbol{b} \tag{5}$$

Para minimizar $C(\boldsymbol{w},\boldsymbol{b})$, sua variação deve ser negativa. Define-se então o gradiente de $C(\boldsymbol{w},\boldsymbol{b})$ na equação (6) onde

$$\nabla C(\boldsymbol{w},\boldsymbol{b}) = \left\langle \frac{\partial C}{\partial \boldsymbol{w}}, \frac{\partial C}{\partial \boldsymbol{b}} \right\rangle \tag{6}$$

Reescrevendo a equação (5) utilizando a equação (6), e reescrevendo $\boldsymbol{w}$ e $\boldsymbol{b}$ como coordenadas de um vetor, obtém-se a equação (7).

$$\Delta C(\boldsymbol{w},\boldsymbol{b}) \approx \nabla C(\boldsymbol{w},\boldsymbol{b}) \cdot \Delta\langle \boldsymbol{w},\boldsymbol{b}\rangle^T \tag{7}$$

Para fazer com que $\Delta C(\boldsymbol{w},\boldsymbol{b})$ seja sempre negativo, o vetor formado por $\boldsymbol{w}$ e $\boldsymbol{b}$ na equação (7) deve ser igual à equação (8).

$$\Delta\langle \boldsymbol{w},\boldsymbol{b}\rangle^T = -\eta \nabla C(\boldsymbol{w},\boldsymbol{b})^T \tag{8}$$

O termo η é chamado de taxa de aprendizagem quando se trata de redes neurais. Substituindo a equação (8) na equação (7), e restringindo a taxa de aprendizagem como sendo sempre maior que zero, é garantida a convergência do algoritmo, já que a variação da função $C(\boldsymbol{w},\boldsymbol{b})$ é sempre menor ou igual a zero, como apresentado na equação (9).

$$\Delta C(\boldsymbol{w},\boldsymbol{b}) \approx \nabla C(\boldsymbol{w},\boldsymbol{b}) \cdot [-\eta \nabla C(\boldsymbol{w},\boldsymbol{b})^T] = -\eta \|\nabla C(\boldsymbol{w},\boldsymbol{b})\|^2 \tag{9}$$

Sendo $\boldsymbol{w}'$e $\boldsymbol{b}'$ como os valores atualizados das variáveis $\boldsymbol{w}$ e $\boldsymbol{b}$, define-se a equação (10).

$$\langle \boldsymbol{w}', \boldsymbol{b}' \rangle^T - \langle \boldsymbol{w}, \boldsymbol{b} \rangle^T = \Delta \langle \boldsymbol{w}, \boldsymbol{b} \rangle^T = -\eta \nabla C(\boldsymbol{w}, \boldsymbol{b})^T \tag{10}$$

Por fim, rearranjando a equação (10), encontra-se o método de atualização das variáveis independentes da função $C(\boldsymbol{w}, \boldsymbol{b})$, representando pela equação (11), de modo a garantir que seu valor nunca aumente caso a equação (8) seja respeitada (isto é, a variação dos pesos e constantes tenda à zero) [Ruder 2017].

$$\langle \boldsymbol{w}', \boldsymbol{b}' \rangle = \langle \boldsymbol{w}, \boldsymbol{b} \rangle - \eta \nabla C(\boldsymbol{w}, \boldsymbol{b}) \tag{11}$$

2.2. Retropropagação do erro

Segundo [Nielsen 2015], para determinar o gradiente da equação (4) é necessário calcular sua derivada parcial em relação a todos os pesos e constantes existentes na rede, para depois atualizá-los com a equação (11). O procedimento para obter o gradiente consiste em propagar diretamente as entradas do conjunto de treinamento, calcular o erro entre a saída da rede e do conjunto de treinamento, utilizando a equação (4), e calcular as derivadas parciais do erro em relação a cada peso e constante da rede. Como as variações dos erros das camadas posteriores influenciam as variações das camadas anteriores, na forma de uma cadeia de derivadas (regra da cadeia), o cálculo é realizado da saída para a entrada, pois as derivadas parciais das camadas posteriores não precisam ser recalculadas. Sendo L o número de camadas da rede neural, $\boldsymbol{a}^l$ o vetor de saída da camada l e $\boldsymbol{z}^l$ o vetor das somas ponderadas de cada camada, define-se a derivada do erro em relação à soma ponderada da camada l pela equação (12).

$$\frac{\partial C}{\partial \boldsymbol{z}^l} = \frac{\partial C}{\partial \boldsymbol{a}^L} \cdot \frac{\partial \boldsymbol{a}^L}{\partial \boldsymbol{z}^L} \cdot \left(\prod_{n=1}^{L-l} \frac{\partial \boldsymbol{z}^{L-n+1}}{\partial \boldsymbol{a}^{L-n}} \cdot \frac{\partial \boldsymbol{a}^{L-n}}{\partial \boldsymbol{z}^{L-n}} \right) \tag{12}$$

As derivadas parciais do erro com relação aos pesos e constantes de cada camada são definidas pelas equações (13) e (14), respectivamente.

$$\frac{\partial C}{\partial \boldsymbol{w}^l} = \frac{\partial C}{\partial \boldsymbol{z}^l} \cdot \frac{\partial \boldsymbol{z}^l}{\partial \boldsymbol{w}^l} \tag{13}$$

$$\frac{\partial C}{\partial \boldsymbol{b}^l} = \frac{\partial C}{\partial \boldsymbol{z}^l} \cdot \frac{\partial \boldsymbol{z}^l}{\partial \boldsymbol{b}^l} \tag{14}$$

O conjunto das derivadas parciais de $C(\boldsymbol{w}, \boldsymbol{b})$ em relação à $\boldsymbol{w}^l$ e $\boldsymbol{b}^l$ para todas as L camadas constituem o gradiente $\nabla C(\boldsymbol{w}, \boldsymbol{b})$ da equação (11).

3. Aplicação do método

Para aplicação do método, foi utilizado um conjunto de dados coletados em um laboratório de testes para motores a combustão. Nesse tipo de laboratório, o motor é instrumentado com diversos sensores, e alguns atuadores são utilizados para fornecer cargas (forças) no motor e simular as condições que ele se encontraria em rotas e ciclos de testes escolhidos por quem realiza o teste. Alguns desses sensores instrumentados em laboratório não estão disponíveis no produto final, sendo necessária a instalação forçada em veículos para a validação em rotas reais (fora do laboratório). O conjunto de dados dispõe de 64000 amostras coletadas de 13 sensores, tais como medidor de torque, velocidade do motor, temperatura e pressão, submetidas a um ciclo de teste *In-Service Conformity*, ciclo esse definido como certas condições físicas, mas escolhido pelo fabricante como uma rota local onde possa ser testado

[União Européia 2009]. Os dados foram todos normalizados por motivos de confidencialidade. A Figura 3 apresenta as 500 primeiras amostras do conjunto de dados.

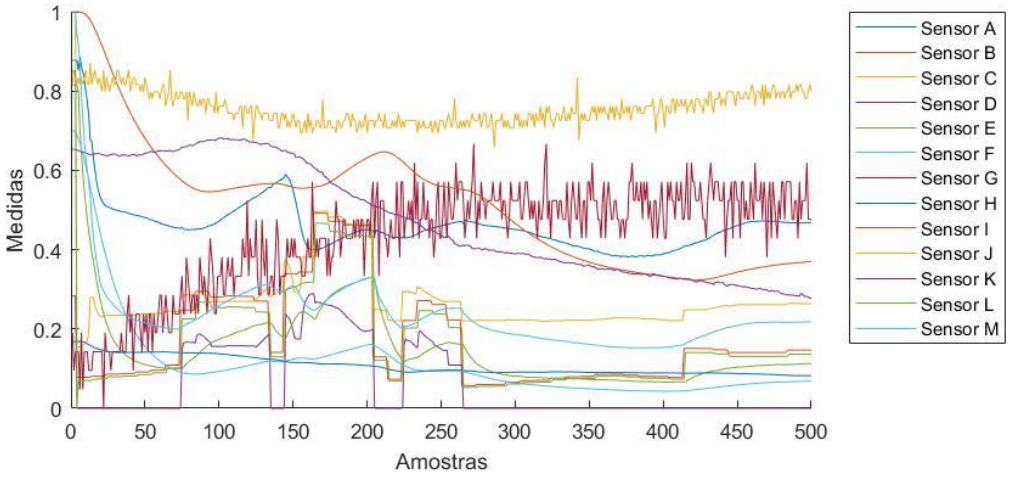


Figura 3 - Conjunto de dados de laboratório – Sensores *onboard*

Esses sensores estão presentes no motor submetido aos testes, sendo alguns deles sensores em *hardware* e outros virtuais, mas nenhum utilizando redes neurais. Outro sensor, que não está presente no motor testado, foi instrumentado em laboratório e é um sensor de pressão não disponível no produto final. As primeiras 500 amostras desse sensor instrumentado, também normalizadas, são apresentadas na Figura 4.

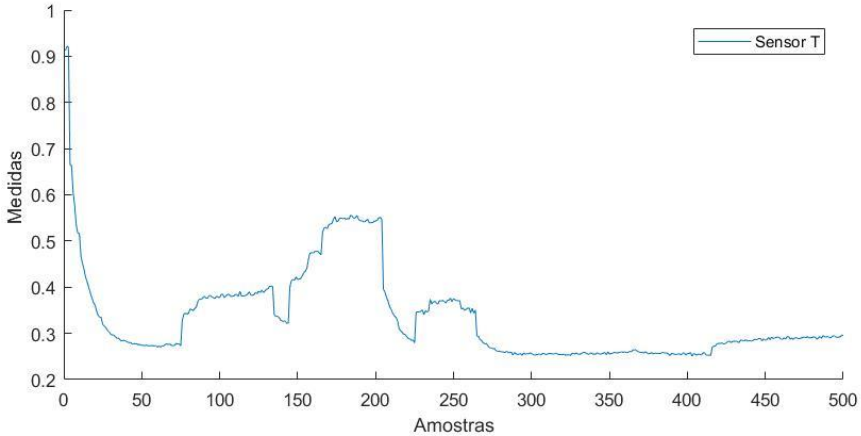


Figura 4 - Conjunto de dados de laboratório – Sensor instrumentado

A topologia de rede neural utilizada foi a mais simples recomendada, sendo o número de neurônios da camada oculta igual ao número de entradas, e o número de neurônios de saída igual ao número de saídas [Goodfellow et al. 2016]. A rede neural artificial adotada, então, possui treze neurônios com função de ativação sigmóide na camada oculta, e um neurônio com função de ativação linear na saída (isto é, a saída é a própria entrada), como apresentado na Figura 5.

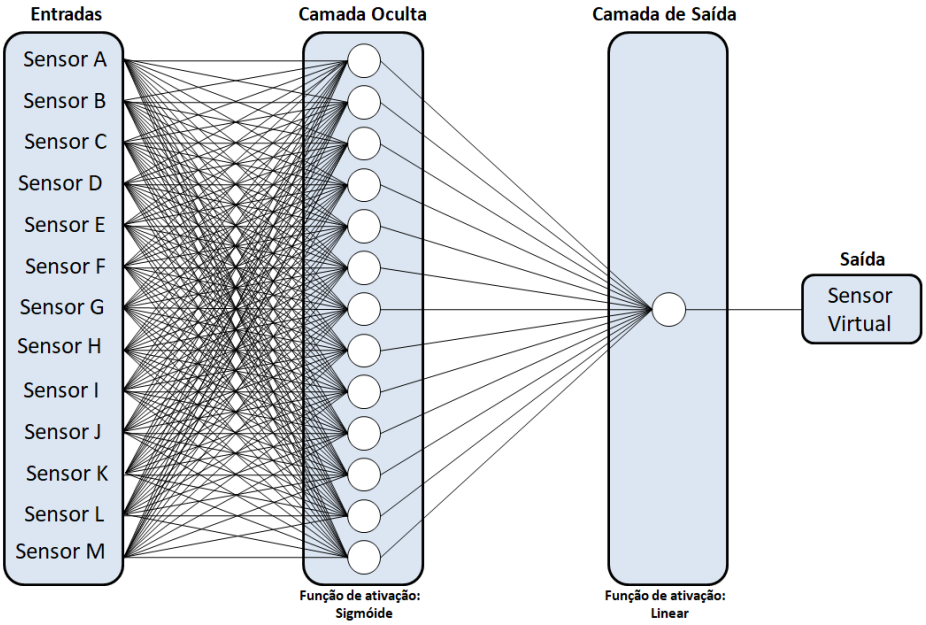


Figura 5 - Topologia da rede neural utilizada

O principal motivo para manter a rede simples é diminuir o custo computacional para embarcá-la em uma ECU (*electronic control unit*), componentes originalmente propostos para melhorar o desempenho de motores, particularmente em termos de eficiência de consumo de combustível. Atualmente, fabricantes têm instalado ECUs não apenas para controle de motores, mas também para realizar várias funções visando segurança e conveniência para o motorista, como o ABS (*anti-lock braking system*) e os assistentes inteligentes de estacionamento. Essas ECUs formam uma rede de comunicação entre si e outros componentes, podendo ser divididas em outras subredes e totalizarem até 70 ECUs em veículos com mais tecnologia embarcada [Choi et al. 2016].

As 64000 amostras, tanto de entrada quanto de saída, foram separadas em três categorias: treinamento, validação e teste, sendo separadas em 32000, 16000 e 16000 amostras, respectivamente. O conjunto de treinamento foi utilizado para atualização dos ganhos, enquanto do conjunto de validação foi utilizado para, ao final de cada iteração (época), verificar se a rede mantém uma boa generalização, diminuindo o erro para um conjunto de dados não contido no conjunto de treinamento. O conjunto de teste foi utilizado para dar o resultado final e verificar se o erro do indicado no treinamento se assemelha ao erro apresentado por um conjunto de dados diferente. A convergência do erro obtida para os três conjuntos, após o treinamento e o teste, é apresentada na Figura 6.

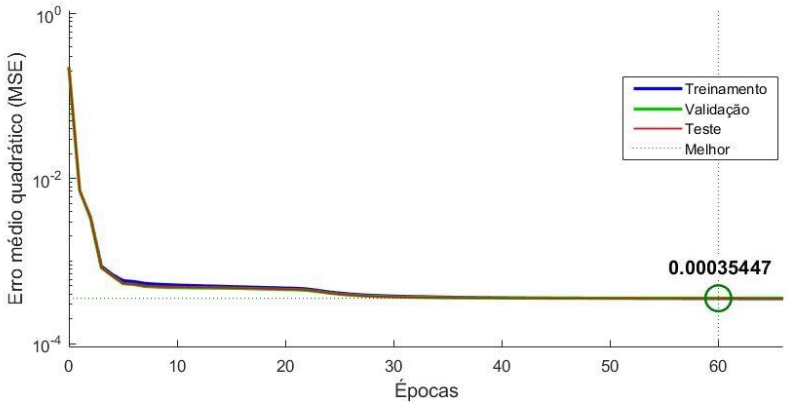


Figura 6 - Convergência do erro para o treinamento

O treinamento se encerrou após 66 iterações, sendo que o menor valor de erro médio quadrático foi obtido na iteração 60. A diferença entre essas duas iteração é a quantidade de testes de validação realizados, ou seja, quando o erro do conjunto de validação para de aumentar, são consideradas seis iterações de margem para verificar se o erro volta a diminuir. Para uma melhor visualização do resultado, a Figura 7.a apresenta a comparação entre a saída da rede e o resultado esperado para as amostras do conjunto de teste, enquanto a Figura 7.b apresenta o erro entre esses dois sinais.

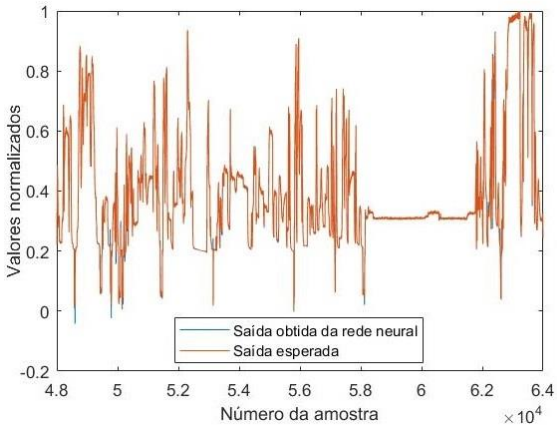


Figura 7.a - Comparação do resultado

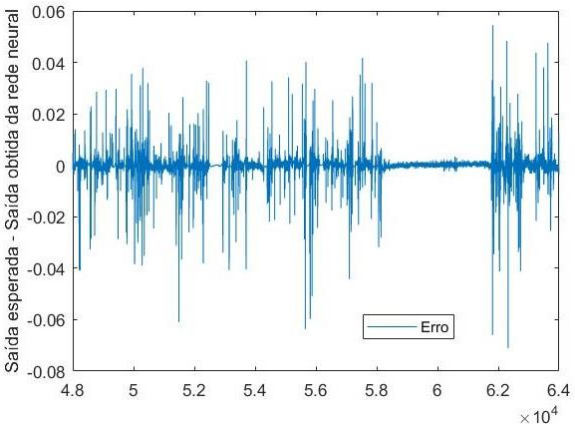


Figura 7.b – Erro entre as saídas

Para comparar o quão próximo os valores obtido e esperado estão, foi utilizado o coeficiente de determinação R^2, que resultou em 0,9933 para o conjunto de teste apresentado na Figura 7 [Quinino et al. 2011]. Comparando o erro absoluto entre as amostras do conjunto de teste, obteve-se o histograma apresentado pela Figura 8, representando uma distribuição com média -0,00000703 e desvio padrão igual a 0,0159.

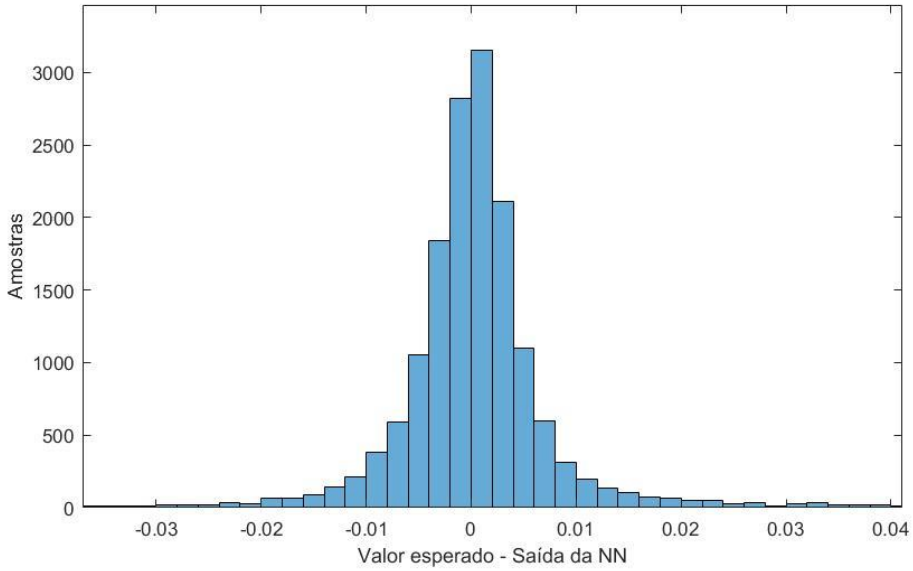


Figura 8 - Histograma de erros absolutos entre o valor esperado e a saída da rede neural.

4. Conclusão

Os resultados obtidos nesse trabalho demonstram que é plausível o uso de redes neurais artificiais de tipo *feed-forward* multicamada como sensores virtuais na indústria automobilística. Considerando a simplicidade topológica da rede neural apresentada e a não utilização do máximo

potencial dos hiper-parâmetros (isto é, inicialização de ganhos, função custo, número de camadas, número de neurônios, etc.), os resultados apresentados demonstram uma alternativa interessante para reduzir custos no melhoramento de produtos.

As principais limitações do método se encontram nos dados, tanto na precisão de aquisição em laboratório quanto na sua quantidade. Quanto ao custo computacional requerido pelas plataformas eletrônicas onde seriam inseridas, as redes neurais, quando alimentadas diretamente, utilizam apenas multiplicações e somas, enquanto as funções de ativação podem ser representadas por tabelas linearizadas, com mais ou menos pontos dependendo da precisão exigida. A quantidade de pesos é que dita a quantidade de memória de programa necessária. Para o caso apresentado nesse artigo, a quantidade de pesos foi de 182, somada a 14 constantes, sendo um peso para cada sinal de entrada, para cada neurônio (isto é, 13 entradas para cada um dos 13 neurônios da primeira camada, somados a 13 pesos entre a camada oculta e a camada de saída), e uma constante para cada neurônio da rede. Se cada um desses valores ocupar 4 bytes, serão necessários 784 bytes para alocar todos os ganhos e constantes da rede, e mais um mapa de linearização para representar a função de ativação sigmóide. Uma alternativa à técnica apresentada nesse artigo, visando a otimização do desempenho computacional, seriam as redes neurais com função de ativação de base radial com operações de ponto flutuante com precisão customizada, proposto por [Ayala et al. 2017].

Para trabalhos futuros, sugere-se a comparação do método apresentado nesse artigo com outras técnicas de aprendizado de máquina (*machine learning*), tais como SVR (*support vector regression machines*) e florestas aleatórias, tanto em desempenho quanto em custo computacional, levando-se em conta as especificações de ECUs. Sugere-se também a comparação para outros conjuntos de dados, com sensores diferentes dos apresentados nesse artigo.

Agradecimentos:

Os autores agradecem o suporte financeiro ao CNPq por meio dos projetos 303908/2015-7-PQ, 404659/2016-0-Univ, 204893/2017-8-PDE e 150501/2017-0-PDJ.

Referências

Ayala, H. V. H., Muñoz, D. M., Llanos, C. H., Coelho, L. S. (2017). Efficient hardware implementation of radial basis function neural network with customized-precision floating-point operations. *Control Engineering Practice*, 60: 124-132.

Choi, W., Jo, H. J., Woo, S., Chun, J. Y., Park, J., Lee, D. H. (2016). Identifying ECUs through Inimitable Characteristics of Signals in Controller Area Networks. *IEEE Transactions on Vehicular Technology*, PP(99): 1-1.

Goodfellow, I., Bengio, Y., Courville, A. (2016). Deep Learning Book. *MIT Press*. Boston, Massachusetts, USA.

Hirsenkorn, N., Hanke, T., Rauch, A., Dehlink, B., Rasshofer, R., Biebl, E. (2016). Virtual sensor models for real-time applications. *Advances in Radio Science*, 14: 31-37.

Morey, B. (2015).Controlling engines with virtual sensors. *SAE International*. Disponível em: http://articles.sae.org/14156/. Acessado em 2018-03-24.

Nielsen, M. A. (2015). Neural networks and deep learning, *Determination Press*. Disponível em: http://neuralnetworksanddeeplearning.com. Acessado em 2018-03-24.

Pathak, D., Halale, V. P. (2017). An introductory approach to virtual sensors and its modelling techniques. *International Journal of Scientific & Engineering Research*, 7: 461-464.

Pinelis, M. (2013). Automotive sensors and electronics: trends and developments in 2013. *Automotive sensors and electronics expo*, 2013, Detroit, Michigan, USA.

Quinino, R., Reis, A., Bessegato, L. F. (2011). O coeficiente de determinação R^2 como instrumento didático para avaliar a utilidade de um modelo de regressão linear múltipla. Artigo apresentado no Programa de Pós-Graduação em Estatística da UFMG. Belo Horizonte.

Rastogi, K., Saini, N. (2017). Virtual sensor modelling using neural networks with coefficient-based adaptive weights and biases search algorithm for diesel engines. Disponível em: https://arxiv.org/ftp/arxiv/papers/1712/1712.08319.pdf. Acessado em 2018-03-25.

Ruder, S. (2017). An overview of gradient descent optimization algorithms. Disponível em: https://arxiv.org/pdf/1609.04747.pdf. Acessado em 2018-03-24.

União Européia (2009). REGULATION (EC) No 595/2009 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 18 June 2009 on type-approval of motor vehicles and engines with respect to emissions from heavy duty vehicles (Euro VI) and on access to vehicle repair and maintenance information and amending Regulation (EC) No 715/2007 and Directive 2007/46/EC and repealing Directives 80/1269/EEC, 2005/55/EC and 2005/78/EC. Disponível em: http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2009:188:0001:0013:EN:PDF. Acessado em 2018-03-24.

Zaharia, C., Clenci, A. (2013). Study on virtual sensors and their automotive applications. *University of Pitesti: Automotive and Transports Department*. Pitesti, Romênia.