XXIII SEMEAD
Seminários em Administração

novembro de 2020
ISSN 2177-3866

APLICAÇÃO DE REDES NEURAIS ARTIFICIAIS PARA PREVISÃO DE DEMANDA DE GASOLINA NO BRASIL

RAFAEL CIMATTI
FACULDADE DE ENGENHARIA DA FUNDAÇÃO ARMANDO ALVARES PENTEADO (FEFAAP)

JORGE LUIZ DE BIAZZI
UNIVERSIDADE DE SÃO PAULO (USP)

APLICAÇÃO DE REDES NEURAIS ARTIFICIAIS PARA PREVISÃO DE DEMANDA DE GASOLINA NO BRASIL

1. INTRODUÇÃO

Existem muitas variáveis que influenciam a demanda de um determinado produto no mercado e há grande dificuldade em lidar com a não linearidade nos métodos matemáticos atualmente empregados. As aplicações de redes neurais artificiais não se limitam a problemas lineares e são particularmente eficientes quando há grande disposição de dados históricos, sendo assim uma alternativa interessante para os problemas de previsão de demanda.

Utilizando-se da mesma tecnologia aplicada à inteligência artificial em diversos sistemas computadorizados, como carros autônomos e assistentes inteligentes, e com um volume de dados adequados, espera-se chegar a um resultado de previsão de melhor qualidade quando comparado aos métodos tradicionais.

A tecnologia de redes neurais artificiais não é novidade e já foi alvo de estudos por mais de uma década; entretanto, devido aos avanços computacionais dos dias de hoje, sua aplicação vem se tornando cada vez mais relevante. Presente atualmente em diversos sistemas no mundo, as redes neurais alimentam a base da inteligência artificial presente, por exemplo, nos carros autônomos: da mesma forma que vamos treinar a rede neural para prever dados de demanda através de exemplos de dados históricos, no carro ela é treinada a desviar de obstáculos com base nos dados dos sensores.

Essa é uma tecnologia muito interessante, pois não se prende à programação mais comum, na qual se devem prever todas as possibilidades em um código e permitir que o computador seja treinado para um fim especifico.

O treinamento da rede, entretanto, depende muito do volume de dados disponível, sendo que sua eficiência está ligada ao ‘aprendizado, por exemplo, dos dados alimentados na entrada. Como essa já é uma característica esperada para problemas de previsão de demanda, espera-se que não seja uma particularidade da rede neural.

Se a aplicação de redes neurais de fato demonstrar desempenho superior, ela poderá ser utilizada por empresas e até mesmo comercializada, lembrando que cada rede deverá ser treinada com um conjunto de dados específicos, o que a torna diferente para cada caso de aplicação.

2. FUNDAMENTAÇÃO TEÓRICA

2.1. Previsão de Demanda

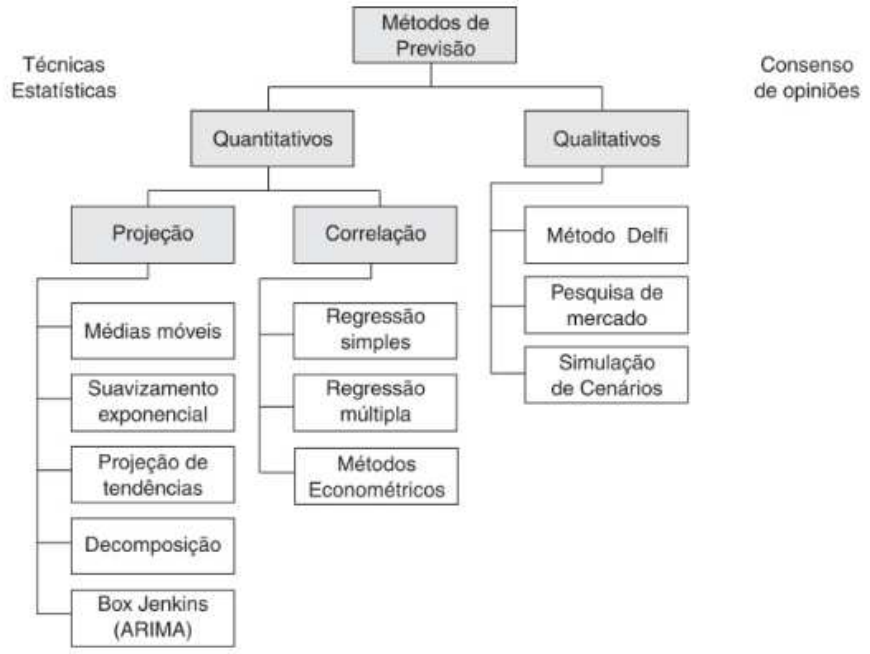
Segundo Mesquita (2011), o conceito de demanda pode ser explicado pela disposição do mercado em consumir determinado produto ou serviço feito por uma empresa. Além disso, a demanda sofre por influência de vários fatores internos e externos à organização. Geralmente, os métodos de previsão de demanda são baseados em históricos de vendas, porém cada método tem uma maneira diferente de tratar esses dados.

“A incerteza da demanda no mercado consumidor propaga-se a montante das cadeias de suprimento” (MESQUITA, 2011). O fenômeno, denominado Efeito Chicote, diz respeito justamente ao encadeamento que ocorre nas cadeias de suprimento, envolvendo desde o cliente no varejo até as fábricas dos fornecedores, e mostra como as incertezas da demanda afetam cada nível.

Segundo Corrêa, Gianesi e Caon (2012), os métodos de previsão de demanda são separados em métodos qualitativos e quantitativos. Os primeiros são baseados em opiniões e julgamentos pessoais. Já nos métodos quantitativos, utilizam-se dados e técnicas estatísticas

para traçar a previsão. Os métodos quantitativos são mais adequados quando existem informações disponíveis. A Figura 1 apresenta uma visão geral dos diferentes métodos tradicionais de previsão de demanda.

Figura 1 - Métodos de previsão



Fonte: Lustosa (2008)

Lee, Song e Mjelde (2008) separam os métodos quantitativos em Projeção de Séries, e Correlação e Regressão. A projeção encara a demanda em função apenas do tempo, assumindo que o comportamento passado é suficiente para prever o que vai acontecer no futuro, segundo Fernandes e Godinho Filho (2010). Já para os métodos de Correlação e Regressão, Lustosa (2008) define que a demanda pode ser relacionada com várias outras variáveis independentes, levando a estimativas de demanda com determinado grau de confiança.

2.1.1. Métodos Quantitativos de Projeção

a) Média móvel: método de projeção em que os últimos "n" períodos são usados para fazer o cálculo da média. Ao adicionar o valor de um período mais recente, o valor mais antigo será descartado, dando origem ao nome "média móvel" (LUSTOSA, 2008);

b) Suavização exponencial simples: segundo Koehler, Snyder e Ord (2011), a suavização exponencial simples parte do pressuposto de que a oscilação da demanda gira em torno de uma constante, sendo apenas corrigida para os diferentes períodos na medida em que novos dados são adicionados;

c) Suavização exponencial com tendência (modelo de Holt): neste método, adiciona-se a variável de tendência que, segundo Mesquita (2011), "reflete o crescimento da demanda de um período para o outro";

d) Suavização exponencial com tendência e sazonalidade (modelo de Holt-Winters): além de contar com a tendência, apresenta também uma variável de sazonalidade que, segundo Lustosa (2008), expressa variações regulares da demanda ao longo de certo ciclo de períodos.

2.1.2. Métodos Quantitativos de Correlação e Regressão

a) Regressão linear simples: os modelos de correlação buscam atrelar alguma variável independente à variável de interesse, neste caso, a demanda. Na regressão simples, apenas uma variável independente é utilizada, segundo Krajewski, Ritzman e Malhotra (2009);

b) Regressão linear múltipla: busca a relação entre múltiplas variáveis independentes e a variável dependente.

2.2. Redes Neurais Artificiais

Nos métodos convencionais de programação, é necessário definir exatamente o que o computador deve fazer, quebrando um grande problema em vários pequenos, que os computadores conseguem solucionar facilmente. Porém, nos métodos que se baseiam em redes neurais, não é necessário definir o que o computador vai fazer, já que o mesmo aprende através de observação dos dados, descobrindo sozinho a solução para o problema, segundo Nielsen (2017).

Apesar de a ideia parecer muito promissora, Winston (2015) mostra que, até recentemente (2006), não havia métodos para treinar as redes neurais de maneira eficiente, o que impedia a solução de problemas mais complexos. Em 2006, foram descobertas técnicas de aprendizado chamadas “deep neural network” ou “aprendizagem profunda”.

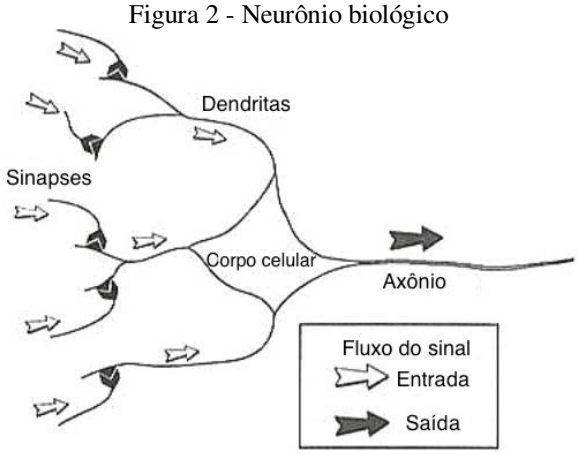
Essas técnicas mudaram o jogo e fizeram com que as redes neurais fossem capazes de resolver problemas importantes e complexos, como reconhecimento de escrita e voz, inteligência artificial para carros e muitos outros que são atualmente utilizados por empresas como Google, Microsoft e Facebook (NIELSEN, 2017).

2.2.1. Conceitos Básicos

Uma rede neural é uma rede interconectada de unidades simples de processamentos e sua funcionalidade é baseada no neurônio de um animal. A capacidade de processamento da rede é armazenada no peso das conexões entre os neurônios, e esses valores são obtidos através do processo de aprendizagem (GURNEY *et. al.*, 1997).

Winston (2015) mostra como é um neurônio humano (temos cerca de 100 bilhões deles). Os neurônios se comunicam por impulsos elétricos e as conexões entre eles são chamadas de sinapses, localizadas nos ramos das células chamadas de dendritos. Cada neurônio recebe centenas de conexões de outros neurônios e, caso os sinais atinjam um certo nível limite, o neurônio “dispara” ou gera um sinal de resposta, transmitindo o impulso para outros neurônios através dos ramos axônio (Figura 2).

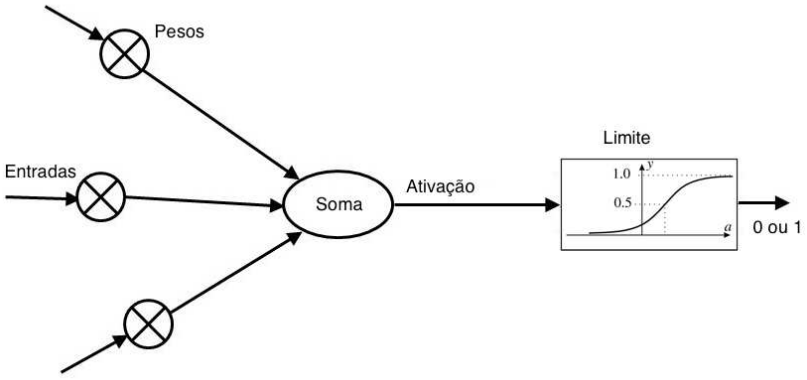
Figura 2 - Neurônio biológico



Fonte: Gurney (1997)

Segundo Gurney *et al.* (1997), o equivalente aos neurônios biológicos nas redes neurais artificias são nós ou unidades; as sinapses são modeladas por um valor único chamado de peso, que multiplica a saída de um nó antes de chegar como uma entrada no próximo nó. Dentro do nó ocorrem funções aritméticas simples e, caso o resultado atinja um certo valor limite, o nó produz um valor de 0 ou 1 (Figura 3).

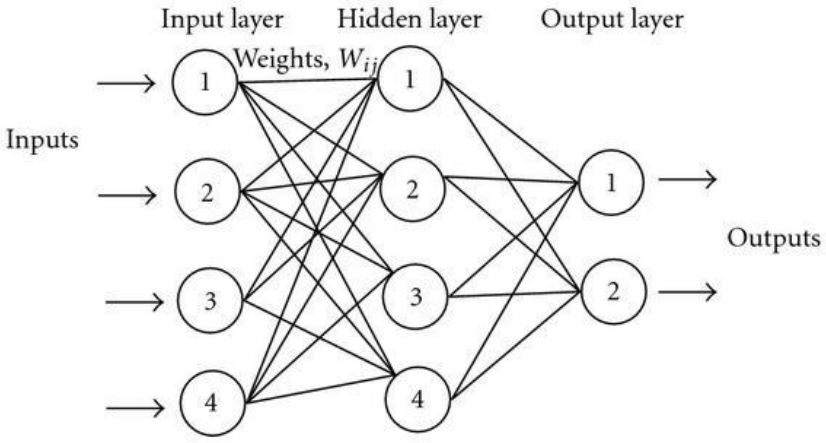
Figura 3 - Neurônio artificial



Fonte; Gurney (1997)

Uma rede neural artificial é composta por várias camadas de nós e suas conexões, que podem também ser chamadas de pesos. Uma rede neural artificial (RNA) deve ter no mínimo 3 camadas: a primeira é chamada de "input layer" ou camada de entrada, a segunda de camada escondida ou "hidden layer", onde ocorre o processamento dos dados de entrada, e por último há a de saída ou "output layer", onde os valores de saída da rede são apresentados. Uma RNA pode ter múltiplas camadas escondidas, o que a torna mais complexa, segundo Nielsen (2017). A Figura 4 apresenta um exemplo de uma RNA com 3 camadas; os pesos são as linhas que conectam os neurônios.

Figura 4 - Rede neural artificial simples



Fonte: http://mlexplore.org/ (acesso em 04/05/2018)

2.2.2. Tipos de Redes Neurais Artificiais (RNA)

Segundo Gurney *et al.* (1997), o aprendizado das RNAs consiste em alterar os valores dos pesos entre os neurônios a fim de se obter o valor desejado na saída. O processo de treinamento trabalha com uma massa de dados de treinamento, na qual temos entradas e

saídas pré-definidas e um programa de computador altera os pesos das conexões entre os neurônios até que a rede consiga, dentro de uma margem de erro aceitável, acertar os valores de saída. Quando isso acontece, os valores dos pesos são armazenados e a rede está treinada, pronta para ser utilizada. O modo como o programa de computador calcula os pesos e altera os valores pode variar entre tipos diferentes de RNAs. Alguns tipos são definidos a seguir.

Feedforward Neural Network:

- Informação se move em apenas uma direção (entrada → saída);
- Tipo mais simples de RNA;
- Não possui ciclos ou *loops*;
- Não possui conceito de tempo.

Recurrent Neural Network:

- Informação se move nas duas direções (entrada → saída e saída → entrada);
- Possui mais métodos de treinamento devido ao fluxo bidirecional da informação;
- Possui conceito de tempo;
- Permite modelos mais complexos para classificação e previsão de série temporal.

2.3. Avaliação de modelos de previsão

Segundo Mesquita (2008), o modo mais básico para avaliar o resultado de uma previsão é simplesmente a diferença entre o valor real e o valor previsto, assumindo que o valor real já é conhecido. Para avaliar o viés do modelo, é possível calcular o Erro Médio (EM), considerando valores reais (Dt) e valores previstos (Ft) ao longo de "n" períodos:

$$EM = \frac{\sum_{t=1}^{n}(Ft - Dt)}{n}$$

Para medir a precisão do modelo de previsão é possível utilizar o Erro Absoluto Médio (EAM) e Erro Quadrático Médio (EQM):

$$EAM = \frac{\sum_{t=1}^{n}|Ft - Dt|}{n}$$

$$EQM = \frac{\sum_{t=1}^{n}(Ft - Dt)^2}{n}$$

3. METODOLOGIA

A metodologia utilizada para este trabalho foi a experimental, por meio da qual se buscou a avaliação e comparação entre um método de previsão mais tradicional (a suavização exponencial) e o baseado em rede neural, com o objetivo de determinar qual método seria mais adequado para o conjunto de dados escolhidos.

Para isso, foi selecionado como histórico de demanda o consumo aparente de gasolina segundo a Agência Nacional do Petróleo (ANP), medido em milhares de barris por dia, entre janeiro de 1979 e junho de 2017 (dados do IPEA - Instituto de Pesquisa Econômica Aplicada). No total, há 462 períodos (meses), dos quais os últimos 12, entre julho de 2016 e junho de 2017, serão utilizados para avaliar os modelos.

3.1. Previsão por suavização exponencial

O método de previsão de demanda tradicional mais adequado para este trabalho e conjunto de dados foi o de projeção, na hipótese de trajetória. A hipótese de sazonalidade foi testada, porém não foi possível identificar um padrão, sendo a hipótese descartada. Além disso, foi utilizada a suavização exponencial, que considera os dados mais recentes do histórico como mais relevantes. A Equação 1 mostra o cálculo feito para a estimativa de projeção com trajetória:

$$E(t) = S(t_0) + (t - t_0) \times R(t_0) \quad (1)$$

Em que: E(t) = estimativa de demanda para período t; t = período atual; t_0 = último período do histórico; $S(t_0)$ = base atual; $R(t_0)$ = tendência atual.

Para a suavização, é necessário adotar um ponto de partida para iniciar o processo de suavização. Neste estudo, foi feita uma regressão linear com os primeiros 36 meses para encontrar os valores de "a" e "b" (o valor da abcissa no ponto de intersecção com o eixo y e a inclinação da reta, respectivamente), utilizando-se o Excel®. Com a regressão, foi possível encontrar os valores iniciais de $S(t_0)$, que é a base da demanda, e o $R(t_0)$, que é a tendência da demanda, utilizando-se as Equações 2 e 3.

$$R(t_0) = b \quad (2)$$

$$S(t_0) = a + b \times t_0 \quad (3)$$

Em que: a = abscissa no ponto de intersecção com o eixo y;b = inclinação da reta.

A previsão foi feita partindo do período 36 até o 462; porém, apenas os últimos 12 períodos foram utilizados para cálculo de erros de previsão e comparação entre os métodos. Além disso, por se tratar da suavização exponencial, os valores dos coeficientes de suavização (alfa e beta) foram iterados manualmente entre 0,1 e 0,3 e também através do Solver a fim de se obter os menores erros. A Equação 4 é utilizada para atualizar os valores de $S(t_0)$, enquanto a Equação 5 serve para atualizar $R(t_0)$, à medida em que as demandas passam a ser conhecidas.

$$S(t_0) = \alpha \times V(t_0) + (1 - \alpha) \times E(t_0) \quad (4)$$

$$R(t_0) = \beta \times [S(t_0) - S(t_0 - 1)] + (1 - \beta) \times R(t_0 - 1) \quad (5)$$

Em que: α, β = coeficientes de suavização; $V(t_0)$ = demanda no último período do histórico (t_0); $E(t_0)$ = estimativa de demanda para t_0.

3.2. Previsão baseado em rede neural

A previsão experimental utilizada nesse trabalho foi baseada no uso de redes neurais, mais especificamente a do tipo *Recurrent Neural Network*, que nos permite mover os dados nas duas direções, treinando a rede neural com os próprios resultados dela, além de possuir o conceito de tempo, que é adequado para uma série temporal como a que temos disponível. Para esse método de previsão, foi necessário o desenvolvimento de um software específico de rede neural, no qual fosse possível ter total controle sobre as variáveis e parâmetros utilizados. As variáveis que poderiam ser modificadas neste método são:

- Número de iterações: a quantidade de vezes que os dados devem ser iterados pela rede neural durante o processo de treinamento com o objetivo de diminuir a diferença entre os valores de previsão e os reais;
- Número de neurônios: a quantidade de neurônios (unidades de processamento) contida na RNA.

O método para encontrar os melhores valores para as variáveis acima foi o de tentativa e erro, sempre comparando os erros para determinar qual seria a melhor combinação entre o número de iterações e neurônios. O diagrama final da rede neural é apresentado na Figura 5.

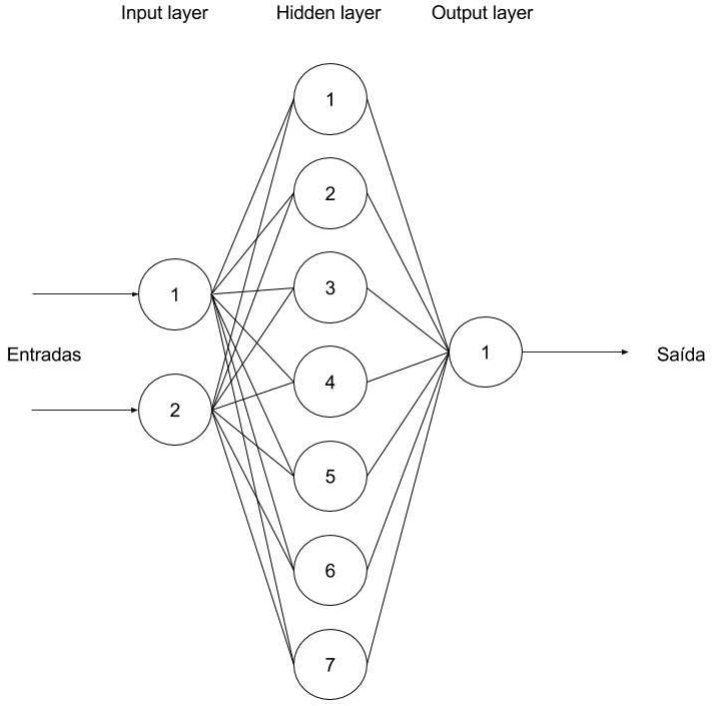
A camada de entrada recebe os valores de data e demanda observada naquela data; por isso, possui dois neurônios. Na camada do meio, chamada de *hidden layer*, acontece o processamento dos dados, e conta com 7 neurônios. Finalmente, na camada de saída temos apenas 1 neurônio, que é o valor da previsão da demanda para o próximo período. Foi necessário rodar 250 iterações dos dados para que a rede neural fosse treinada o suficiente a ponto de retornar valores satisfatórios.

Cada ligação entre os neurônios acima possui um valor de peso que é alterado a cada iteração percorrida pela rede neural. O valor de peso é multiplicado pela saída do neurônio na camada anterior (nesse caso, os neurônios na camada de entrada). Para cada um dos sete

neurônios na camada de processamento, os valores obtidos da multiplicação do peso e da entrada do dado são somados e em seguida passa por uma função de ativação antes de ser enviado para a próxima camada. A função de ativação utilizada nesse trabalho foi a tangente hiperbólica. Dependendo do resultado da função de ativação, o dado é propagado para a próxima camada (nesse caso, a camada de saída).

Para cada processo de iteração da RNA, os valores de peso mudam e produzem valores diferentes na camada de saída. Com esses resultados de previsão, calcula-se o EQM (com base nos últimos 12 meses) baseado nos dados originais e, para a próxima iteração, outros valores de peso serão utilizados com a intenção de minimizar o EQM.

Figura 5 - Diagrama da rede neural final.



Fonte: Elaborado pelos autores

Além disso, para esse método de previsão, é necessário tratar os dados de entrada para que a RNA consiga trabalhar corretamente com esses valores. A Figura 6 mostra as primeiras linhas do arquivo de dados de entrada no formato CSV.

Figura 6 - Exemplo de dados CSV.

```
1   "", Sales
2   "1979-01",258.0
3   "1979-02",227.0
4   "1979-03",236.0
5   "1979-04",228.0
6   "1979-05",269.0
7   "1979-06",229.0
8   "1979-07",258.0
9   "1979-08",260.0
10  "1979-09",197.0
11  "1979-10",218.0
12  "1979-11",245.0
13  "1979-12",196.0
```

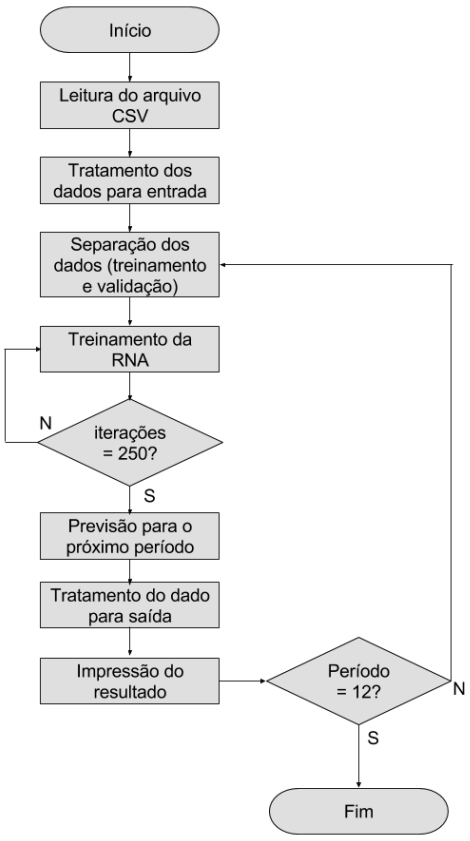
Fonte: Elaborado pelos autores

Os passos executados pelo programa de rede neural para adequar os dados e seguir com o treinamento da RNA são:

1. Leitura do arquivo CSV e análise dos dados;
2. Transformação dos dados em entradas (x) e saídas (y): Para o problema de série temporal, foram utilizados o último valor (t-1) como entrada e o valor atual (t) como saída.
3. Transformação dos dados para remoção de tendência: Os dados de consumo aparente de gasolina parecem aumentar ao longo do tempo, o que significa que a demanda tem a tendência de aumentar ao longo do tempo. Para remover essa tendência, o método mais comum é diferenciar os dados, ou seja, calcular e utilizar a diferença entre o período anterior e o próximo. A operação inversa será executada na saída do dado para adicionar novamente a tendência e calcular a previsão.
4. Para treinar a RNA, é necessário escalar os dados entre -1 e 1, pois essa é a variação que a função de ativação (tangente hiperbólica) suporta. No final do programa, para calcular a previsão final, foi executada a operação inversa.
5. Os dados são divididos entre dados de treinamento e dados de validação. Os dados de treinamento foram os valores de demanda para os primeiros 450 períodos, enquanto que os dados de validação são os últimos 12. A cada previsão feita, o programa adiciona o valor real da demanda do período previsto como um dado de treinamento para a previsão do próximo mês.

O programa para tratamento de dados, criação e treinamento da rede neural e previsão dos dados baseado na rede neural foi desenvolvido exclusivamente para esse trabalho na linguagem Python®, com o fluxograma apresentado na Figura 7. O código-fonte está disponível no Apêndice A. A execução do programa levou 1 hora e 17 minutos, num computador com processador Intel Core i7 2,8GHz, com 16GB de memória RAM.

Figura 7 - Fluxograma do programa da rede neural.



Fonte: Elaborado pelos autores

4. RESULTADOS E DISCUSSÃO

A Tabela 1 apresenta os resultados do uso da técnica de suavização exponencial na hipótese de trajetória, com coeficientes de suavização que minimizam o erro quadrático médio (EQM). Os valores dos coeficientes alfa e beta encontrados são, respectivamente, 0,15 e 0,52.

Tabela 1 - Estimativa exponencial com coeficientes que minimizam EQM

Data	Período	Demanda	Estimativa exponencial	$S(t_0)$	$R(t_0)$	Erro	Erro absoluto	Erro quadrático
2016.07	451	510,00	539,78	535,30	7,01	29,78	29,78	886,92
2016.08	452	526,00	542,32	539,86	5,74	16,32	16,32	266,20
2016.09	453	549,00	545,61	546,12	6,01	-3,39	3,39	11,52
2016.10	454	536,00	552,13	549,70	4,75	16,13	16,13	260,03
2016.11	455	567,00	554,45	556,34	5,73	-12,55	12,55	157,38
2016.12	456	625,00	562,07	571,54	10,62	-62,93	62,93	3959,94
2017.01	457	551,00	582,16	577,47	8,20	31,16	31,16	970,95
2017.02	458	582,00	585,67	585,12	7,91	3,67	3,67	13,50
2017.03	459	585,00	593,04	591,83	7,29	8,04	8,04	64,58
2017.04	460	559,00	599,12	593,08	4,17	40,12	40,12	1609,42
2017.05	461	560,00	597,25	591,65	1,27	37,25	37,25	1387,87
2017.06	462	576,00	592,92	590,38	-0,04	16,92	16,92	286,44
					Médias	10,04	23,19	822,90
alfa=	0,15	beta=	0,52			EM	EAM	EQM

Fonte: Elaborado pelos autores

O critério de escolha dos melhores coeficientes foi a minimização do EQM, pois é uma medida que tende a punir grandes diferenças entre a demanda real e a previsão. Pode-se observar que o erro médio foi de 10,04, o erro absoluto médio foi de 23,19 e o erro quadrático médio foi 822,90.

Os resultados obtidos pelo método de rede neural dependem dos parâmetros de números de neurônios na rede e o número de iterações que a rede neural usa para seu processo de aprendizagem. Esse processo de escolha dos parâmetros é feito na base da tentativa e erro. A Tabela 2 apresenta os erros de previsão obtidos em relação aos parâmetros selecionados, e destaca os parâmetros ideais encontrados que foram utilizados para a previsão final da rede neural, cujos resultados seguem na Tabela 3. Esta revela que os erros para a previsão por rede neural foram de -2,46, 18,17 e 734,51 para erro médio, erro absoluto médio e erro quadrático médio, respectivamente.

As médias de erro para os dois métodos de previsão foram baixas; porém, o método baseado em redes neurais destacou-se ao apresentar erros menores. O Gráfico 1 apresenta as demandas e as previsões feitas pelas duas técnicas.

Tabela 2 - Parâmetros da rede neural

Iterações	Neurônios	EAM	RMSE (*Root Mean Squared Error*)	EQM
100	4	23,852	31,341	982,258
100	9	23,069	30,704	942,736
60	3	21,593	30,035	902,101
60	5	22,264	29,505	870,545
60	9	19,264	28,626	819,448
60	7	18,838	28,497	812,079
60	4	20,079	28,495	811,965
60	3	21,362	28,357	804,119
100	9	19,482	27,770	771,173
100	7	19,117	27,727	768,787
60	7	19,076	27,610	762,312
60	7	19,063	27,505	756,525
60	9	18,562	27,369	749,062
100	4	20,082	27,223	741,092
60	9	18,526	27,113	735,115
100	7	18,683	26,988	728,352
100	7	20,320	26,810	718,776
60	9	19,777	26,702	712,997
60	7	19,273	26,658	710,649
150	7	18,567	26,507	702,621
150	7	18,091	25,424	646,380
250	7	16,292	24,776	613,850

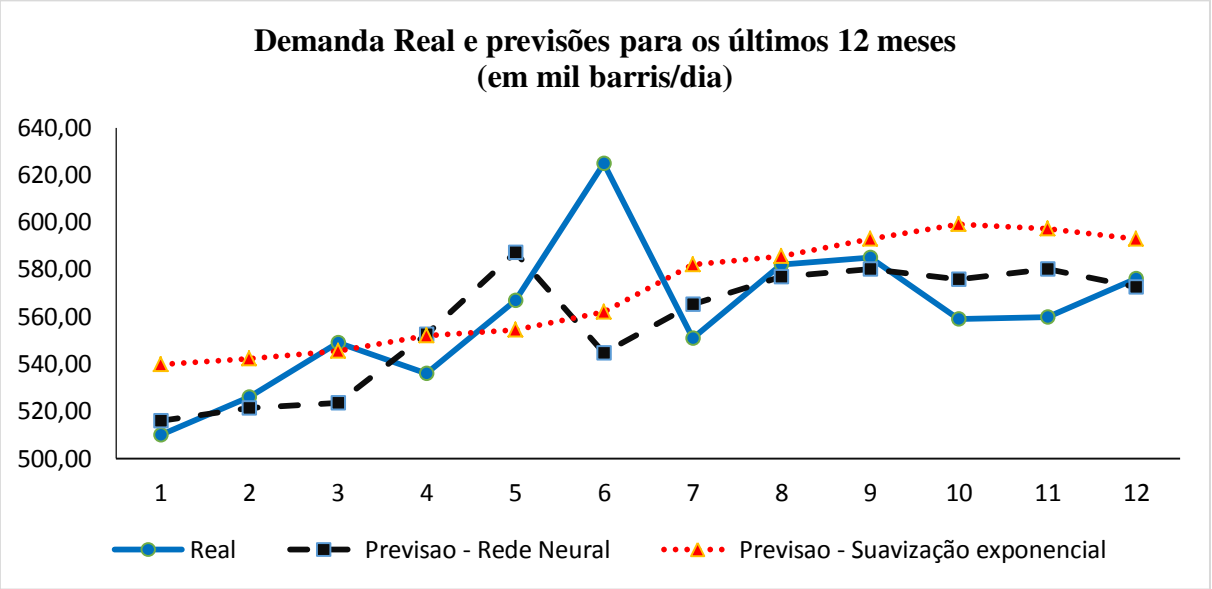
Fonte: Elaborado pelos autores

Tabela 3 - Estimativa por rede neural

Período	Demanda	Previsão	Erro	Erro Absoluto	Erro quadrático
451	510,00	516,02	6,02	6,02	36,22
452	526,00	521,50	-4,50	4,50	20,24
453	549,00	523,44	-25,56	25,56	653,45
454	536,00	552,48	16,48	16,48	271,58
455	567,00	587,31	20,31	20,31	412,33
456	625,00	544,65	-80,35	80,35	6456,91
457	551,00	565,23	14,23	14,23	202,56
458	582,00	576,97	-5,03	5,03	25,26
459	585,00	580,08	-4,92	4,92	24,24
460	559,00	575,97	16,97	16,97	287,87
461	560,00	580,29	20,29	20,29	411,63
462	576,00	572,56	-3,44	3,44	11,80
		Médias	-2,46	18,17	734,51
			EM	EAM	EQM

Fonte: Elaborado pelos autores

Gráfico 1 - Gráfico da demanda real e previsões por rede neural e suavização exponencial.



Fonte: Elaborado pelos autores

O método de previsão de demanda baseado em redes neurais obteve resultados superiores quando comparados com o método de previsão baseado em suavização exponencial. Os cálculos apresentados na Tabela 4 revelam que os valores de demanda projetados pela rede neural resultaram em um erro absoluto médio e erro quadrático médio menores em 22% e 11%, respectivamente, em relação aos erros obtidos pela suavização exponencial.

Tabela 4 – Resumo da comparação entre as técnicas.

Método	EM	EAM	EQM
Rede Neural	-2,46	18,17	734,51
Suavização Exponencial (α=0,15; β=0,52)	10,04	23,19	822,90
Ganho (%)	76%	22%	11%

Fonte: Elaborado pelos autores

Vale ressaltar que foram buscados os coeficientes alfa e beta para minimizar o EQM; caso se desejasse minimizar o EAM, este possivelmente seria menor. O mesmo é verdade para a otimização da rede neural, em que o EQM foi novamente utilizado como função-objetivo.

A vantagem do método de suavização exponencial é sua característica de reprodutibilidade, sendo possível aplicar o mesmo método com apenas algumas alterações e esforço relativamente baixo para adequar a previsão para o histórico de outro produto. Já para o método baseado em redes neurais, a vantagem foi a obtenção de resultados mais precisos, embora mais tendenciosos (e com viés negativo, o que pode ser um problema caso a subestimação traga custos maiores do que a superestimação da demanda).

Entretanto, há um ponto importante que deve ser notado no que diz respeito ao esforço e nível de customização necessário para o método baseado em redes neurais. Para esse trabalho, por exemplo, foi necessário desenvolver um programa especificamente para esse conjunto de dados. Claro que o programa pode ser reutilizado, mas os valores ideais para os parâmetros de número de iterações e número de neurônios teriam de ser encontrados

novamente. Além disso, outra dificuldade desse método é a falta de aplicações prontas disponíveis, o que torna o acesso ao método mais difícil.

A tendência é que esse tipo de método de previsão de demanda seja empregado pelas empresas pelo alto desempenho, porém de maneira manual e em muitas vezes com conhecimento e esforço internos, pelo menos até que o método seja padronizado e disponibilizado no mercado.

REFERÊNCIAS BIBLIOGRÁFICAS

CORRÊA, Henrique Luiz; GIANESI, Irineu Gustavo Nogueira; CAON, Mauro. **Planejamento, programação e controle da produção: MRP II/ERP conceitos, uso e implantação: base para SAP, Oracle Applications e outros softwares integrados de gestão.** São Paulo: Atlas, 2012.

FERNANDES, Flavio Cesar Faria; GODINHO FILHO, Moacir. **Planejamento e controle da produção dos fundamentos ao essencial**. São Paulo: Editora Atlas, 2010.

GURNEY, Kevin. **An Introduction to Neural Networks**. University of Sheffield, 1997.

IPEA – Instituto de Pesquisa Econômica Aplicada. **Consumo aparente – gasolina – média – qde./dia**. Rio de Janeiro, 2017. Acesso em: 26 Set, 2017.

KOEHLER, A. B.; SNYDER, R. D.; ORD, J. K. **Forecasting Models and Prediction Intervals for the Multiplicative Holt-Winters Method. International Journal of Forecasting**, v. 17, n.2, p.269-286, 2001.

KRAJEWSKI, L. J.; RITZMAN, L. P.; MALHOTRA, M. **Administração da produção e operações**. 8. ed. São Paulo: Pearson Prentice Hall, 2009.

LEE, C. K.; SONG, H. J.; MJELDE, J. W. **The forecasting of international expo tourism using quantitative and qualitative techniques**. Tourism Management, v. 29, n. 6, p. 1084-1098, 2008.

LUSTOSA, Leonardo; MESQUITA, Marco A.; QUELHAS, Osvaldo; OLIVEIRA, Rodrigo. **Planejamento e controle da Produção**. Rio de Janeiro: Elsevier, 2008.

NIELSEN, Michael. **Neural Networks and Deep Learning**. Maio, 2017. Disponível em: <http://neuralnetworksanddeeplearning.com/>. Acesso em 12 Jun, 2017.

WINSTON, Patrick Henry. **Artificial Intelligence**. Massachusetts Institute of Technology, 2015. Disponível em: <https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-034-artificial-intelligence-fall-2010/index.htm>. Acesso em: 12 Jun, 2017.

Apêndice A – Código fonte do programa de rede neural

```
import os
import time
import warnings
from time import gmtime, strftime
from pandas import DataFrame
from pandas import Series
from pandas import concat
from pandas import read_csv
```

```
from pandas import datetime
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import MinMaxScaler
from keras.models import Sequential
from keras.layers import Dense, Activation, Dropout
from keras.layers import LSTM
from math import sqrt
from matplotlib import pyplot
import numpy
from keras import optimizers

os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3' #Hide TensorFlow warnings
warnings.filterwarnings("ignore") #Hide Numpy warnings

# date-time parsing function for loading the dataset
def parser(x):
	return datetime.strptime(x, '%Y-%m')

# frame a sequence as a supervised learning problem
def timeseries_to_supervised(data, lag=1):
	df = DataFrame(data)
	columns = [df.shift(i) for i in range(1, lag+1)]
	columns.append(df)
	df = concat(columns, axis=1)
	df.fillna(0, inplace=True)
	return df

# create a differenced series
def difference(dataset, interval=1):
	diff = list()
	for i in range(interval, len(dataset)):
		value = dataset[i] - dataset[i - interval]
		diff.append(value)
	return Series(diff)

# invert differenced value
def inverse_difference(history, yhat, interval=1):
	return yhat + history[-interval]

# scale train and test data to [-1, 1]
def scale(train, test):
	# fit scaler
	scaler = MinMaxScaler(feature_range=(-1, 1))
	scaler = scaler.fit(train)
	# transform train
	train = train.reshape(train.shape[0], train.shape[1])
	train_scaled = scaler.transform(train)
	# transform test
	test = test.reshape(test.shape[0], test.shape[1])
	test_scaled = scaler.transform(test)
```

```
        return scaler, train_scaled, test_scaled

# inverse scaling for a forecasted value
def invert_scale(scaler, X, value):
        new_row = [x for x in X] + [value]
        array = numpy.array(new_row)
        array = array.reshape(1, len(array))
        inverted = scaler.inverse_transform(array)
        return inverted[0, -1]

# fit an LSTM network to training data
def fit_lstm(train, batch_size, nb_epoch, neurons):
        X, y = train[:, 0:-1], train[:, -1]
        X = X.reshape(X.shape[0], 1, X.shape[1])
        model = Sequential()
        model.add(LSTM(neurons, batch_input_shape=(batch_size, X.shape[1],
X.shape[2]), stateful=True))
        model.add(Dense(1))
        model.add(Activation('tanh'))
        model.compile(loss='mean_squared_error', optimizer='rmsprop')

        for i in range(nb_epoch):
                model.fit(X, y, epochs=250, batch_size=batch_size, verbose=2,
shuffle=False)
                model.reset_states()
        return model

# make a one-step forecast
def forecast_lstm(model, batch_size, X):
        X = X.reshape(1, 1, len(X))
        yhat = model.predict(X, batch_size=batch_size)
        return yhat[0,0]

def previsaoRNA(trainingSize, supervised_values):
        # split data into train and test-sets
        train, test = supervised_values[0:-trainingSize], supervised_values[-trainingSize:]
        print('%d períodos de treinamento e %d de teste' % (len(train), len(test)))

        # transform the scale of the data
        scaler, train_scaled, test_scaled = scale(train, test)

        # fit the model
        lstm_model = fit_lstm(train_scaled, 1, 1, 7) #param2 é o batch_size, param3 é o
número de epochs, param4 é o número de neurons!

        # forecast the entire training dataset to build up state for forecasting
        train_reshaped = train_scaled[:, 0].reshape(len(train_scaled), 1, 1)
        lstm_model.predict(train_reshaped, batch_size=1)

        # make one-step forecast
```

```
        X, y = test_scaled[0, 0:-1], test_scaled[0, -1]
        yhat = forecast_lstm(lstm_model, 1, X)
        # invert scaling
        yhat = invert_scale(scaler, X, yhat)
        # invert differencing
        yhat = inverse_difference(raw_values, yhat, len(test_scaled)+1)
        expected = raw_values[len(train) + 1]
        print('Mês=%d, Previsão=%f, Esperado=%f' % (13-trainingSize, yhat, expected))
        return 1

# Começo
print("Começando em: ", strftime("%Y-%m-%d %H:%M:%S", gmtime()))

# Fix random seed for reproducibility
numpy.random.seed(7)

# load dataset
series = read_csv('data.csv', header=0, parse_dates=[0], index_col=0, squeeze=True,
date_parser=parser)
# transform data to be stationary
raw_values = series.values
diff_values = difference(raw_values, 1)

# transform data to be supervised learning
supervised = timeseries_to_supervised(diff_values, 1)
supervised_values = supervised.values

for i in range(12, 0, -1): #Ajuste para fazer previsão mês-a-mês e incluir último mês que já
passou:
        previsao = previsaoRNA(i, supervised_values)
print("Finalizado em: ", strftime("%Y-%m-%d %H:%M:%S", gmtime()))
```