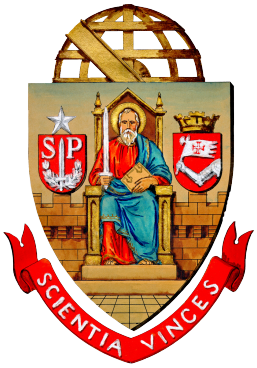
Universidade de São Paulo

Escola de Engenharia de São Carlos

Trabalho de Conclusão de Curso

Revisão Bibliográfica

Pedro Lino Falcão - 10309009



2021

São Carlos - SP

Contents

1. Introdução

Para o desenvolvimento desta tese, será necessário identificar os principais tipos de redes neurais, suas vantagens e desvantagens e, principalmente, como aplicá-las em escoamentos bifásicos, para a melhora na produção de óleo e gás.

Para cobrir essa lacuna, serão respondidas quatro perguntas principais:

1. Explicar e equacionar o funcionamento de uma feedforward neural network e recurrent neural network
2. Quais as diferenças entre FNN e RNN, vantagens e deficiências de cada modelo?
3. Explicar e equacionar uma long short-term memory (LSTM), como ela soluciona problemas inerentes da RNN
4. Como LSTM pode ser aplicada em escoamento bifásico (foco industria óleo e gás)?

2. Neural Networks

2.1 Artificial Neural Networks - ANN

Artificial Neural Networks são modelos de computação baseados no conhecimento de neurociência, imitando o funcionamento do cérebro, com neurônios artificiais, normalmente referidos por nódulos, (*nodes* ou *units*), e suas respectivas sinapses, que são representadas pelas conexões entre nódulos [5].

Esses nódulos guardam valores, que podem representar valores de entrada, de saída ou, na camada escondida - ou seja, na região de decisão da rede neural - valores a serem computados, essa computação é feita por funções matemáticas e operações com matrizes, como descrito por [2].

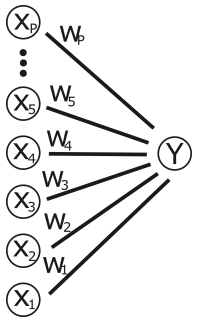


Figure 2.1: Exemplo de um nódulo, recebendo entrada de outros p nódulos, cada qual com seus respectivos pesos.

Pode-se descrever um nódulo genérico, como o proposto na Figura 2.1, pela seguinte equação:

$$y = f(\sum_{j=1}^{p} w_j \times x_j + x_0)$$

Seja y o valor que um nódulo posterior irá receber, x_j, com j variando entre 1 e p, são os nódulos na camada anterior, w_j são os pesos de cada conexão, demonstrando o grau de influência que aquela conexão possui na decisão final e x_0 é o termo de viés de cada conexão. Após a realização das operações, o número real resultante é passado pela função *f*, denominada *função de ativação*, que irá normalizar os valores, dentro de uma faixa de valores predeterminada. [5] sugere que escolhas comuns para a função de ativação são a função sigmoide, $\sigma(z)$, e a função de tangente hiperbólica, $tanh(z)$:

$$\sigma(z) = \frac{1}{1+e^{-z}} \qquad\qquad tanh(z) = \frac{e^{z}-e^{-z}}{e^{z}+e^{-z}}$$

Redes neurais são hoje usadas em diversas áreas de conhecimento que se beneficiam de algoritmos para o reconhecimento de padrões, como problemas de optimização e de robótica (principalmente aqueles que interagem com humanos e precisam ser capazes de reconhecer fala e emoções) [2]. Em especial, neste artigo será destacado o uso de ANNs como **medidores de escoamento virtuais** (*Virtual Flow Meter* - VFM), que têm se mostrado uma estratégia bastante promissora para o monitoramento de escoamentos, como sugere [1].

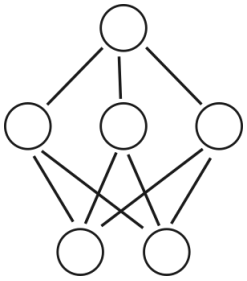


Figure 2.2: Representação de uma ANN genérica, com dois nódulos de entrada, uma camada escondida com três nódulos e uma última camada com um único nódulo

Quando se trata da forma como o algoritmo realiza seus cálculos, pode-se sugerir duas classes principais: **com ciclos** e **sem ciclos** [3]. Aquelas redes neurais que não possuem ciclos, ou seja, que a computação é feita em ordem, da camada mais baixa até a mais alta, são as Feedforward Neural Networks (FNNs), enquanto que aquelas cujos nódulos de uma mesma camada, ou até anteriores, se comunicam, ou seja, formam ciclos, são as Reccurent Neural Networks (RNNs) [5].

2.2 Feedforward Neural Networks - FNN

Como sugerido anteriormente, as redes neurais do tipo Feedforward não apresentam ciclos, veja Figura 2.2, a computação é feita da ordem das camadas mais baixas até a a mais alta. A otimização deste algoritmo vem da alteração dos pesos, a fim de de diminuir a distância entre o resultado conseguido e o resultado esperado, [5].

Um dos algoritmos mais populares para essa otimização é o de *Backpropagation*, proposto por [8]. No período de treino se definem entradas cujas saídas são conhecidas, de forma que se possa avaliar o desempenho da rede neural, em termos da diferença entre o esperado e o real e, assim, derivar essa função a fim de encontrar o valor onde a diferença é mínima.

O funcionamento geral de uma feedforward se dá pela computação de baixo para cima, como explicado na Seção 2.1, e, posteriormente pela realização do **Gradiente Descendente** da função que calcula a diferença

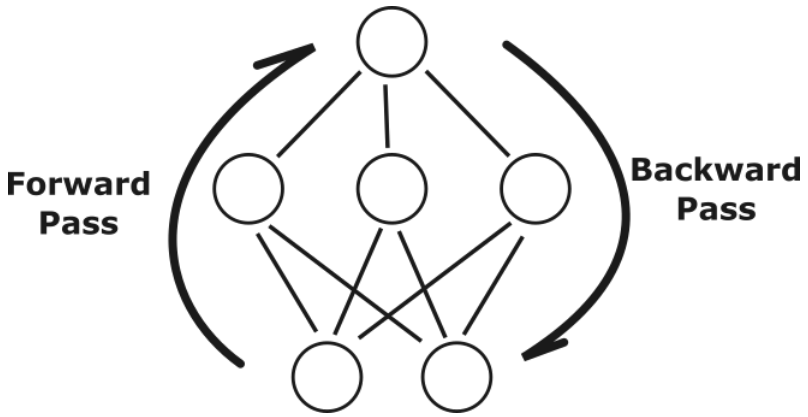


Figure 2.3: Representação de uma FNN, com forward passa indo de baixo para cima e o backward pass, de cima para baixo, otimizando os valores de cada peso

entre o resultado esperado em um nódulo e o resultado obtido, em termos dos pesos, obtendo assim o mínimo da função, esse algoritmo de *backpropagation* é então computado de cima para baixo, daí o nome(Figura 2.3). A partir do funcionamento da Equação 2.1, seja o resultado esperado, y_j^e, e a função de erro entre o valor esperado e o obtido, E(y_j, y_j^e), seja também:

$$a_j = \sum_{j=1}^{p} w_j \times x_j + x_0$$

Então, define-se, para a camada final:

$$\delta_k = \frac{\Delta E(y_k, y_k^e)}{\Delta y_k^e} \times f'(a_j)$$

Onde f'(a_j) é a derivada primeira da função de ativação. A partir deste resultado, pode-se escrever para a camada imediatamente anterior:

$$\delta_j = f'(a_j) \times \sum_{j} \delta_k \times w_{jk}$$

De forma que, por fim, se possa calcular o gradiente descendente da função erro em cada nódulo por:

$$\frac{\partial E(y_k, y_k^e)}{\partial w_{jj'}} = \delta_j \times v_{j'}$$

A partir dessas equações e da definição da constante de aprendizado, η, os pesos de cada conexão se atualizam, e os mecanismos de *forward pass* e *backward pass* acontecem de forma intercalada, até que se encontre o mínimo, local ou global, da função [8]. As FNN são amplamente utilizadas em aplicações de reconhecimento de padrões, porém, quando se tratando de aplicações sequenciais, ou seja, que levam em conta o efeito do tempo, elas

rapidamente demonstram suas limitações, por serem capazes de produzir saídas somente a partir das entradas, sem que entradas anteriores sejam consideradas, [3].

2.3 Recurrent Neural Networks - RNN

A adição de ciclos às redes neurais supre o problema do tempo, uma vez que permite que as decisões sejam embasadas não apenas na entrada em determinado tempo t, mas também no histórico de computação [3]. As Recurrent Neural Networks são as redes neurais com esse tipo de arquitetura.

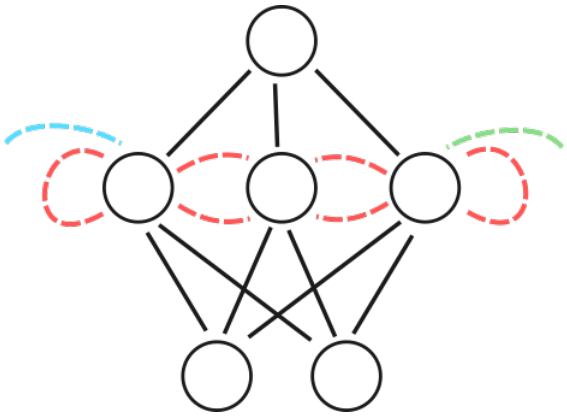


Figure 2.4: Uma arquitetura de RNN, as linhas tracejadas em vermelho representam os ciclos e as linhas tracejada azul e verde representa conexões com o tempo imediatamente antes (t-1) e o imediatamente depois(t+1), respectivamente

Além da adição de ciclos, outra mudança das RNNs em relação às FNNs é que as entradas e saídas dela são sequências, que são séries de vetores, que demonstram uma certa evolução, podendo ser temporais ou não, mas, num contexto geral, sequências são aplicadas quando se busca uma noção de ordem (i.e. primeiro, segundo) [5].

A forma como os estados anteriores é contabilizado, é pela adição aos resultados no instante anterior (Uma representação disso pode ser vista na Figura 2.4), também de forma ponderada, de forma que, como sugerido por [3], para o instante de tempo, t, tem-se que o valor de um nódulo é dado pelo valor recebido pelos nódulos abaixo, bem como o de outros nódulos na própria camada no instante, *t-1*:

$$b_j^t = f(\sum_{k=1}^{p} w_j k \times x_k^t + x_0 + \sum_{j=1}^{J} w_{jj} \times b_j^{t-1})$$

Onde o nódulo b_j, no instante t, recebe os valores da camada inferior, também no instante t, da mesma forma que em FNNs (Veja a Equação 2.1), bem como da mesma camada, no instante *t-1*, formando assim um ciclo. O resultado passa então pela função de ativação, f, que pode ser a mesma utilizada nas FNNs ([3]).

Nota-se que para o instante inicial, *t=1*, deve-se escolher um valor arbitrário para b_j^{t-1}, [3] sugere que o valor pode ser definido como nulo, porém, que existem benefícios em relação à robustez do sistema em escolher valores não nulos.

A otimização de uma RNN se dá, também, pelo mecanismo de *Backpropagation*, porém com a atualização deste para que leve em conta a progressão de tempo. [5] sugere que o método de *Backpropagation through time*, BPTT, é o mecanismo mais utilizado nas RNNs para a atualização dos pesos.

Como sugere [9], o mecanismo de Backpropagation through time é bastante parecido com o de Backpropagation, pura e simplesmentes, a diferença é que não apenas se faz necessário ir das camadas superiores até as inferiores, mas também de tempos futuros para tempos passados, o equacionamento, como demonstra [3], será apresentado a seguir. Seja δ_k representa a derivada da função erro pelo valor dos nódulos e a_h^t:

$$a_h^t = \sum_{k=1}^{p} w_j k \times x_k^t + x_0 + \sum_{j=1}^{J} w_{jj} \times b_j^{t-1}$$

Então pode-se definir:

$$\delta_h^t = f'(a_h^t) \times \sum_{k=1}^{K} w_h k \times \delta_k^t + x_0 + \sum_{h=1}^{H} w_{hh'} \times \delta_{h'}^{t-1}$$

Por fim, pela regra da cadeia das derivadas, da mesma forma como o mecanismo normal de backpropagation, tem-se:

$$\frac{\partial E(y_k^t, y_k^e, t)}{\partial w_{ij}} = \sum_{t=1}^{T} \delta_j^t \times b_i^t$$

Esse valor é então multiplicado pela taxa de aprendizado e é usado para atualizar os pesos e otimizar a acurácia do algoritmo.

3. Long Short-Term Memory - LSTM

Afim de resolver problemas inatos aos outros tipos de RNNs, principalmente quanto a propagação de erro, ao longo da predição, o que faz com que os resultados tendam a explodir ou implodir [3], e dificuldades de treinamento [5], foi desenvolvida uma nova arquitetura de RNN, a *Long Short-Term Memory* [4].

A **Célula de Memória** é a estrutura principal desse tipo de rede neural, ela é composta por portões lógicos e substitui os nódulos usados nos modelos anteriores - Essa é justamente a razão pela qual o gradiente se comporta, mostrando a superioridade desse tipo de ANN ao longo de tempo [4].

A arquitetura da célula de memória é caracterizada pelas seguintes estruturas, descritas por [5]:

- Nódulo de entrada **g**: Recebe os valores da camada anterior.
- Portão de entrada **i**: Os portões lógicos são estruturas binárias que permitem, ou não, a passagem do valor para o próximo estágio.
- Estado interno **s**: Essa estrutura leva um erro constante através do tempo, realizando a propagação da constante de erro ao longo do tempo.
- Portão de esquecimento **f**: Essa estrutura foi desenvolvida após o artigo original, mas se mostrou importante para a otimização do algoritmo.
- Portão de saída o_c: De forma semelhante ao portão de entrada, o portão de saída atua sob o estado interno para produzir o resultado final, v_c.

Dessa forma, uma célula de memória em alto nível poderia ser representada pelas equações:

$$s^t = g^t \cdot i^t + f^t \cdot s^{t-1} \qquad\qquad v^t = o^t \cdot s^t$$

É feita também uma representação gráfica dessa célula a seguir, na Figura 3.1:

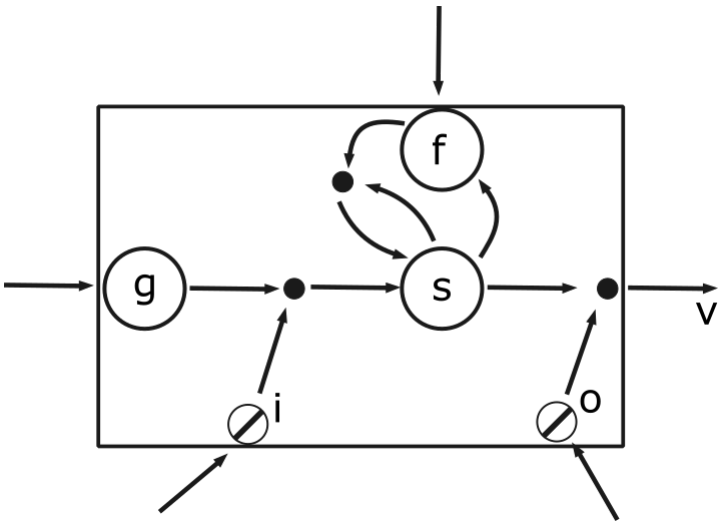


Figure 3.1: Os portões permitem que o algoritmo decida quais informações são relevantes a ponto de serem "lembrados" e passadas para os próximos nódulos, bem como propagadas através do tempo.

Embora existam outras adaptações possíveis para LSTM, que foram desenvolvidas em trabalhos posteriores, o equacionamento original, proposto por [4], descreve a célula de memória (c) de número j - ou seja c_j - da seguinte forma: São três entradas, y^{in_j}, y^{out_j} e net_{c_j} e uma saída, y^{c_j}.

As entradas são definidas pelas seguintes equações:

$$y^{in_j}(t) = \sigma_{in_j}(\sum_u w_{in_j u} \times y^u(t-1))$$
$$y^{out_j}(t) = \sigma_{out_j}(\sum_u w_{out_j u} \times y^u(t-1))$$
$$net_{c_j}(t) = \sum_u w_{c_j u} \times y^u(t-1)$$

O índice u indica a origem do valor, podendo representar unidades de entrada, portões, outras células de memória ou até mesmo a mesma célula (caso onde ocorre recorrência). W representa os pesos dados para cada conexão e σ é a função sigmoide já descrita anteriormente.

A saída da célula de memória é definida por:

$$y^{c_j}(t) = y^{out_j}(t) \times \sigma_{-11}(s_{c_j}(t))$$

Onde σ_{-11} é uma função diferenciável que multiplica os estados internos, s_{c_j}, e também uma variação de σ, para que a saída possua um valor entre -1 e 1:

$$\sigma_{-11}(x) = \frac{2}{1 + e^{-x}} - 1$$

Esses estados internos são denotados por:

$$s_{c_j}(t) = s_{c_j}(t-1) + y^{in_j}(t) \times \sigma_{-22}(net_{c_j}(t))$$

Note que, para o caso inicial, ou seja t = 0, se faz necessário a escolha de um valor arbitrário.É convencionado usar:

$$s_{c_j}(0) = 0$$

A função σ_{-22} é, também, uma variação de σ, para que os valores da saída estejam entre -2 e 2:

$$\sigma_{-22}(x) = \frac{4}{1 + e^{-x}} - 2$$

4. Neural Networks aplicadas à Mecânica dos Fluídos

O uso de softwares para a medição de escoamentos é uma alternativa relativamente barata e confiável para reduzir custos operacionais em plataformas de extração de petróleo *offshore*. Com o uso de instrumentação simples, como medidores de temperatura e pressão, até em condições desfavoráveis ou configurações *menos-que-ideais* de sensoreamento, como descreve [6].

Essa técnica de aferir, a partir de dados indiretos (i.e. informação da temperatura em pontos esparsos), o estado do escoamento, é denominada medição virtual, no caso, *Virtual Flow Meter* - VFM. VFMs podem ser hidrodinâmicos, ou seja, que se baseiam nas propriedades físicas do escoamento e geométricas do poço, ou movidas por dados, como modelos de *Machine Learning* [1].

Há também a possibilidade da junção dessas duas tecnologias, com o chamado *Physics informed Machine Learning*, que combina modelos físicos para auxiliar a predição em situações com quantidade escassa de dados. Essa técnica já se mostrava promissora em outras áreas do conhecimento, mas [7] demonstrou sua qualidade também para a predição de escoamento bifásicos.

Uma vantagem interessante que as VFMs baseadas em dados possuem sob as hidrodinâmicas é a capacidade de predição a longos períodos, particularmente daquelas baseadas no algoritmo de LSTM, enquanto os modelos hidrodinâmicos possuem uma capacidade de predição fixa.[1] obteve resultados promissores de predição de escoamentos usando apenas sensores de temperatura e pressão e dados com ruído.

Utilizando, por tanto, uma rede neural do tipo LSTM em conjunto com um modelo físico de escoamentos, como o apresentado por [7], espera-se obter ótimos resultados, capazes de auxiliar diversas indústrias com predições do estado futuro do escoamento, a partir de sensores relativamente baratos. Fazendo com que, para casos onde há poucos dados, o modelo físico preencha a lacuna, enquanto que, no caso contrário, a rede neural seja capaz de se destacar.

Essa tecnologia tem potencial de gerar bons resultados financeiros, por permitir o melhor dimensionamento de estruturas de interação com o escoamento, aumentando assim o tempo de vida destes, e até socio-ambientais, se for levada em conta a capacidade desse tipo de sistema de prevenir acidentes.

Bibliography

[1] Nikolai Andrianov. A machine learning approach for virtual flow metering and forecasting. *IFAC-PapersOnLine*, 51(8):191–196, 2018.

[2] Bing Cheng and D. M. Titterington. Neural networks: A review from a statistical perspective. *Statistical Science*, 9(1):2–30, 1994.

[3] Alex Graves. Supervised sequence labelling with recurrent neural networks. 2012. *https://www.cs.toronto.edu/~graves/phd.pdf*, 2012.

[4] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 11 1997.

[5] Zachary C Lipton, John Berkowitz, and Charles Elkan. A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv: 1506.00019* *https://arxiv.org/pdf/1506.00019.pdf*, 2015.

[6] Hallgeir Melbø, Svein Arne Morud, Robert van der Geest, Bjørn Bringeda, and Kjetil Stenersen. Software That Enables Flow Metering of Well Rates With Long Tiebacks and With Limited or Inaccurate Instrumentation. All Days, 05 2003. OTC-15363-MS.

[7] André Mendes Quintino, Davi Lôtfi Lavôr Navarro da Rocha, Roberto Fonseca Júnior, and Oscar Mauricio Hernandez Rodriguez. Flow Pattern Transition in Pipes Using Data-Driven and Physics-Informed Machine Learning. *Journal of Fluids Engineering*, 143(3), 11 2020. 031401.

[8] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[9] P.J. Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.