DAS Departamento de Automação e Sistemas
CTC Centro Tecnológico
UFSC Universidade Federal de Santa Catarina

Uso de Redes Neurais para obtenção de Modelos de Predição para Controladores Preditivos Lineares

Relatório submetido à Universidade Federal de Santa Catarina
como requisito para a aprovação da disciplina:
DAS 5511: Projeto de Fim de Curso

Ricardo Fileti Marcon

Florianópolis, Dezembro de 2019

Uso de Redes Neurais para obtenção de Modelos de Predição para Controladores Preditivos Lineares

Ricardo Fileti Marcon

Esta monografia foi julgada no contexto da disciplina
DAS 5511: Projeto de Fim de Curso
e aprovada na sua forma final pelo
Curso de Engenharia de Controle e Automação

Prof. Julio Elias Normey Rico

Banca Examinadora:

Gabriel Bruzaca Cavalcante/Scopi
Carolina Maia Vettorazzo/Scopi
Orientadores na Empresa

Prof. Julio Elias Normey Rico
Orientador no Curso

Prof. Marcelo de Lellis Costa de Oliveira
Responsável pela disciplina

Me. Marcelo Menezes Morato, Avaliador

Vinicius Heck Peiter, Debatedor

Nicholas Wagner, Debatedor

Resumo

Na indústria de petróleo e gás, controladores preditivos baseados em modelos (MPC) multivariáveis podem ser utilizados para otimizar os processos de produção e de refino. Esta estratégia de controle requer modelos que representem a dinâmica dos processos. Muitos algoritmos MPC utilizam modelos lineares invariantes no tempo em sua formulação, porém estes modelos representam bem o sistema apenas perto de um ponto de operação para o qual foram obtidos. Por outro lado, modelos fenomenológicos não lineares, baseados em equações dinâmicas, podem ser difíceis de obter e normalmente apresentam um alto custo computacional. Uma alternativa é a modelagem baseada em dados, capaz de gerar boas aproximações com um custo computacional reduzido para a execução. Uma das possíveis técnicas para obtenção desse tipo de modelo é a identificação por redes neurais. Entretanto, existem dificuldades na escolha dos parâmetros destas redes e possíveis problemas causados pela respectiva má escolha destes. Estes problemas serão apresentados ao longo do documento, que detalhará algumas técnicas para redução/anulação dos efeitos indesejáveis da má escolha dos parâmetros. O objetivo deste trabalho é obter um modelo não linear através do uso de redes neurais, capaz de representar a dinâmica do sistema em diversos pontos de operação com a possibilidade de ser utilizado, então, para a construção do modelo linear para predição do MPC.

Palavras-chave: Redes Neurais. Identificação. MPC. Modelagem Baseada em dados.

Abstract

To optimize production and refining processes, oil and gas facilities often use multivariable model-based predictive controllers (MPC). This control strategy requires models that represent the dynamics of the processes. Many MPC algorithms use linear time-invariant models in their formulation, but these models only represent the system accurately near the operating point for which they were obtained. On the other hand, nonlinear phenomenological models based on ordinary differential equations can be difficult to obtain and usually have a high computational cost. An alternative is the data-based modeling, capable of generating good approximations with a reduced computational cost for execution. One of the possible techniques for obtaining this type of model is the identification by neural networks. However, there are difficulties in choosing the network's parameters and possible problems caused by their poor choice. These problems will be presented throughout this document, which will detail some techniques for reducing/nullifying the undesirable effects of poor parameter selection. The objective of this work is to obtain a nonlinear model through the use of neural networks, capable of representing the system dynamics in several operating points with the possibility of being used, then, to construct the linear prediction model for the MPC algorithm.

Keywords: Neural Networks. Identification. MPC. Data-based Modeling.

Lista de ilustrações

Lista de tabelas

Sumário

1 Introdução

Uma estratégia de controle aplicada com sucesso na indústria do petróleo e gás é o Controle Preditivo baseado em Modelo (MPC, do inglês *Model Predictive Control*) [1]. Em [2] e [3], os autores propõem o uso de algoritmos MPC para manter sistemas de compressão operando em regiões estáveis, além de minimizar a energia consumida no processo. A cada iteração, o MPC prevê a saída do sistema através de um modelo explícito do mesmo e calcula as ações de controle através da minimização de uma função objetivo, considerando as restrições do sistema. Além disso, o MPC lida facilmente com atrasos e sistemas multivariáveis e ainda permite a inclusão de perturbações medidas na sua formulação.

Para o correto funcionamento do MPC, o modelo do sistema a ser controlado é essencial. O desempenho do MPC está diretamente relacionado a qualidade do modelo utilizado para predição. Caso esse modelo de predição não represente bem o comportamento do sistema, o algoritmo controlador terá dificuldade em atingir os objetivos especificados. Diversas estratégias de controle MPC aplicadas na indústria utilizam modelos de predição lineares com sucesso, mas a medida que o sistema se afasta do ponto de operação no qual o modelo linear foi obtido, esse modelo perde a capacidade de representar o sistema. O *Practical Nonlinear MPC* (PNMPC), proposto por [1], apresenta um MPC que utiliza modelos de predição linearizados sucessivamente a cada iteração do controle, obtidos a partir de um modelo não linear do processo.

Realizar linearizações sucessivas de um modelo não linear a cada iteração do controle pode ser um problema, afinal nem todos os processos possuem modelos dinâmicos fenomenológicos não lineares que representem bem o sistema. Além disso, o custo computacional para resolver modelos fenomenológicos de complexidade elevada pode ser alto demais, tornando muitas vezes inviável seu uso a cada iteração do algoritmo de controle.

Nesse contexto, os modelos baseados em dados estão sendo cada vez mais utilizados para prever as saídas futuras de um sistema [4]. Com uma boa coleta de dados, é possível gerar modelos com funcionamento relativamente simples, com custo computacional mais baixo que modelos baseados na resolução de um conjunto de equações diferencias e ainda assim representando bem o comportamento do sistema em seus diversos pontos de operação. Assim, a obtenção do modelo não linear que representa o sistema é simplificada e o custo computacional reduzido para seu uso o torna viável para aplicações onde o tempo disponível para a simulação do processo é curto.

No contexto deste trabalho, a ferramenta escolhida para a obtenção de modelos a partir de dados foram as redes neurais. Sua escolha se deu pela existência prévia de projetos e desenvolvimentos na empresa utilizando a ferramenta e, portanto, havendo uma maior familiaridade com a aplicação desta quando comparada a outras técnicas de

identificação de sistemas a partir de dados.

Redes neurais são poderosas ferramentas de aprendizado de máquina utilizadas para, através de aprendizado por reforço, reconhecer e aprender a representar ou classificar padrões presentes em conjuntos de dados. Suas aplicações são possíveis nas mais diversas áreas, englobando medicina, economia, planejamento urbano, engenharia e muitas outras, auxiliando em tarefas como, por exemplo, o diagnóstico e predição de doenças, reconhecimento de fala ou escrita manual, *Data Mining* etc.

No contexto de sistemas dinâmicos, as redes neurais apresentam um grande potencial para obtenção de modelos baseados em dados, visto sua capacidade e poder computacional de representar as diversas não linearidades e relações complexas presentes nos sistemas. Uma vantagem bastante atrativa é a versatilidade de aplicação das redes, uma vez que a técnica utilizada não necessita de alterações quando aplicadas a sistemas SISO (*Single Input Single Output*) ou MIMO (*Multiple Input Multiple Output*).

1.1 Objetivo geral

Dada a necessidade de obtenção de modelos não lineares possíveis de se utilizar em conjunto com as técnicas de MPC, este trabalho tem como objetivo desenvolver uma ferramenta de identificação de sistema utilizando redes neurais. Esta ferramenta deverá, ainda, contar com algumas melhorias de modo a simplificar a tarefa de obter modelos capazes de representar com qualidade sistemas dinâmicos não lineares em suas diferentes faixas de operação e que possam ser utilizados como base para obtenção dos modelos lineares de predição do MPC.

Objetivos Específicos

- Desenvolver uma ferramenta capaz de aprender e, posteriormente, representar as dinâmicas de um sistema não linear;
- Aplicar técnicas de taxa de aprendizado adaptativa na ferramenta;
- Aplicar técnicas de seleção de estrutura na ferramenta

1.2 Estrutura do documento

Este trabalho é dividido em seis capítulos:

No capítulo 1, é apresentada a introdução do documento, destacando pontos como apresentação do problema, contextualização e objetivos.

No Capítulo 2, é apresentada a fundamentação teórica com relação às redes neurais artificiais.

No capítulo 3, é apresentada a fundamentação teórica com relação a identificação de sistemas.

No capítulo 4, é apresentado a metodologia utilizada no projeto.

No capítulo 5, é apresentado o sistema utilizado como caso de estudo, onde foram aplicadas e validadas as técnicas vistas neste trabalho, assim como os resultados obtidos após realização dos testes.

No capítulo 6, são apresentadas as conclusões e perspectivas para trabalhos futuros.

2 Redes Neurais

Nas seções a seguir são apresentados alguns conceitos básicos sobre o funcionamento das redes neurais utilizadas neste trabalho.

2.1 *Feedforward Neural Networks*

As redes neurais do tipo *feedforward* são redes cujas conexões não formam ciclos, sendo o cálculo de suas saídas feito exclusivamente em um único sentido: das entradas para as saídas. Sendo assim, essa arquitetura de rede não possui memória, não havendo, então, nenhuma influência das entradas/saídas passadas ou futuras da rede no cálculo das saídas atuais. Um exemplo de FNN é visto na figura 1.

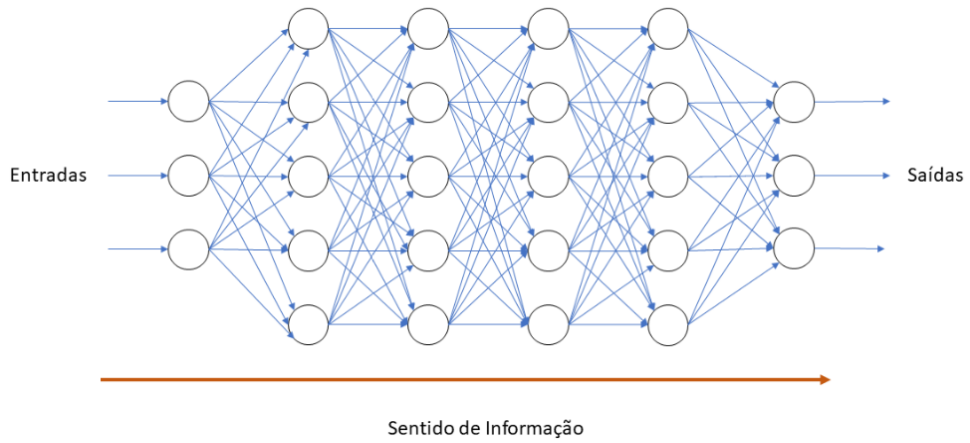

Figura 1 – Rede neural do tipo FNN, onde o fluxo da informação ocorre sempre das entradas para as saídas, sem formação de ciclos.

Conforme é discutido em [5], redes *feedforward* são as mais simples e conhecidas, tendo sido a primeira classe de redes neurais artificiais a ser criada. Dentre as arquiteturas pertencentes a esta classe estão: *Adaline* (*Adaptive Linear Element*), MLP (*Multilayer Perceptron*) e, de forma geral, qualquer arquitetura montada de modo a formar um grafo acíclico dirigido.

Multilayer Perceptron

De acordo com [5, 6], redes do tipo MLP são formadas por ao menos três camadas: camada de entrada, camada(s) oculta(s), e camada de saída. Com exceção da camada de entrada, todas as outras são formadas por neurônios dotados de uma função de ativação e conectados aos neurônios da camada anterior e também aos neurônios da camada seguinte, não havendo conexões entre os neurônios de uma mesma camada. Entre a camada

de entrada e a camada de saída da rede, podem existir uma ou várias camadas ocultas, possuindo, cada uma delas, um ou mais neurônios. Um exemplo de rede MLP é apresentado na Figura 2.

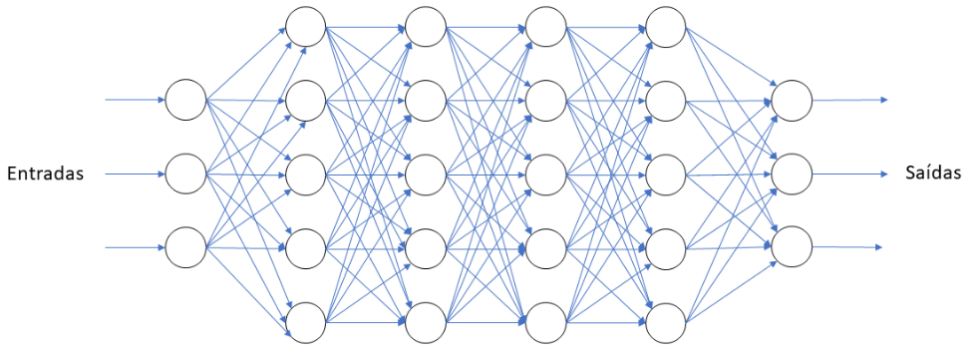

Figura 2 – Rede do tipo MLP com 3 entradas, 3 saídas e 4 camadas ocultas, com 5 neurônios cada.

Normalmente, cada neurônio da rede recebe, através de suas entradas, informações de todos os neurônios da camada anterior e fornece, através de sua saída, informações para todos os neurônios existentes na camada seguinte da rede, caracterizando, assim, camadas completamente conectadas.

2.2 Funções de ativação

Como visto em [5], as funções de ativação desempenham o papel de permitir o mapeamento e aprendizado de relações não lineares, visto que sem a presença dessas, a rede acabaria por ser apenas um modelo de regressão linear. Para tanto, são aplicadas funções não lineares nas saídas de cada um dos neurônios.

O cálculo da saída y_j de cada um dos neurônios da rede passa a ser feito através da função de ativação $f(x_j)$ que recebe como entrada x_j, a soma de todas as entradas daquele neurônio multiplicadas por seus respectivos pesos, assim como representado pela Equação (2.1).

$$x_j = \sum_{i=0}^{n} y_i w_{ij} \tag{2.1}$$

Na Equação (2.1), y_i é a saída do neurônio de índice i da camada anterior, conectado a entrada do neurônio de índice j da camada atual para qual a saída y_j está sendo calculada, w_{ij} é o peso conectando a saída do neurônio de índice i da camada anterior à entrada do neurônio de índice j da camada atual e n é o número total de neurônios que se conectam às entradas do neurônio de índice j da camada atual.

O cálculo da saída y_j do neurônio é feito, então, conforme a Equação (2.2), onde y_j é o valor de saída do neurônio e x_j é o valor obtido através da Equação (2.1) e $f(x_j)$ é a função de ativação definida para aquele neurônio.

$$y_j = f(x_j) \tag{2.2}$$

Qualquer função pode ser utilizada como função de ativação dos neurônios. Porém, deve-se levar em conta que funções demasiadamente complexas ou com elevado custo computacional devem ser evitadas, visto que podem dificultar, ou até mesmo impossibilitar, o aprendizado das redes. Além disso, funções com alto custo computacional podem impactar não somente o treinamento, mas também o tempo de execução das redes. A seguir, serão apresentadas algumas funções de ativação largamente utilizadas e com bom desempenho geral.

Função Sigmoide

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.3}$$

Uma função não linear e continuamente diferenciável, com funcionamento fácil de entender e implementar. Sua saída varia na faixa entre 0 e 1, sendo centrada em 0,5 (Fig. 3). Em alguns casos não é desejável seu uso devido ao fato de gerar apenas valores positivos. Ainda, por não ser uma função centrada em zero, pode trazer uma maior dificuldade para o aprendizado da rede em alguns casos.

Existe ainda o problema da dissipação do gradiente (*"The Vanishing Gradient Problem"*), onde os gradientes propagados durante o treinamento de uma rede vão sofrendo multiplicações por valores menores que 1 a cada camada da rede atravessada, chegando nas camadas iniciais da rede com valores ínfimos. Isso faz com que o ajuste dos pesos, calculado a cada iteração do treinamento da rede, sejam, também, ínfimos. Assim, torna-se muito custoso o treinamento das primeiras camadas da rede, necessitando de uma quantidade muito elevada de iterações para um ajuste minimamente relevante.

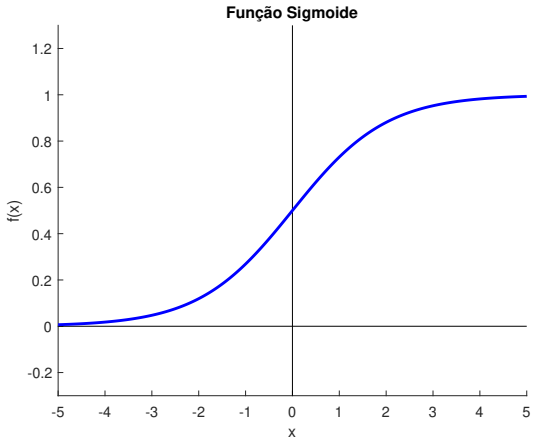

Figura 3 – Representação gráfica da função sigmoide.

Função Tangente Hiperbólica

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2.4}$$

A tangente hiperbólica acaba por ser uma função muito semelhante a sigmoide, tendo as mesmas características funcionais, exceto por suas saídas variarem entre -1 e 1, tendo então as vantagens de ser capaz de assumir valores negativos e ter seu centro em zero. Ainda assim, a função também é sensível ao problema de dissipação do gradiente.

A representação gráfica da função é vista na Figura 4.

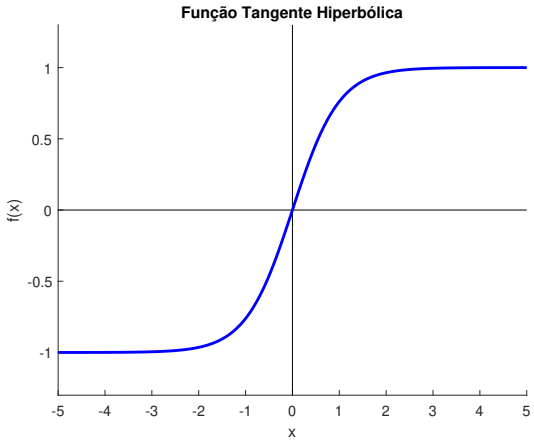

Figura 4 – Representação gráfica da função tangente hiperbólica

Função ReLU

$$f(x) = max(x, 0) \tag{2.5}$$

Ou,

$$f(x) = \begin{cases} 0 \text{ se } x < 0 \\ x \text{ se } x \geqslant 0 \end{cases} \tag{2.6}$$

Uma função não linear e computacionalmente menos custosa que funções como a sigmoide, que utilizam a função exponencial em sua formulação. Por não haver progressivas multiplicações do gradiente por valores menores que 1, a função não é afetada pelo problema da dissipação do gradiente.

Apesar de resolver o problema da dissipação do gradiente, um novo problema surge, visto que para valores abaixo de zero o valor de sua saída, assim como o de seu gradiente, é zero, não sendo possível um ajuste de pesos e propagação de gradiente nesses casos. Assim, é possível que em algum momento do treinamento da rede, o neurônio seja levado para uma região em que jamais é ativado novamente e por consequência tem seu valor de gradiente, a partir daquele momento, sempre zero. Isso faz com que o neurônio torne-se incapaz de aprender, passando a ser considerado um neurônio morto. Este problema recebe o nome de *"Dying ReLU"*.

Para solucionar este novo problema que surge ao se fazer uso da ReLU (Fig. 5), foram criadas variações da ReLU como a *Leaky Rectified Linear Unit* (LReLU, ou *Leaky ReLU*), a *Exponential Linear Unit* (ELU), dentre outras. Estas variações buscam modificar os valores da função para entradas abaixo de zero. Resumidamente, os valores negativos que, ao serem computados pela ReLU, retornavam sempre o valor zero, nessas variantes da função retornam um valor diferente de zero. Assim, mesmo para esses valores a função de ativação possui um gradiente de valor diferente de zero, tornando possível o cálculo de ajustes para os pesos das conexões do neurônio.

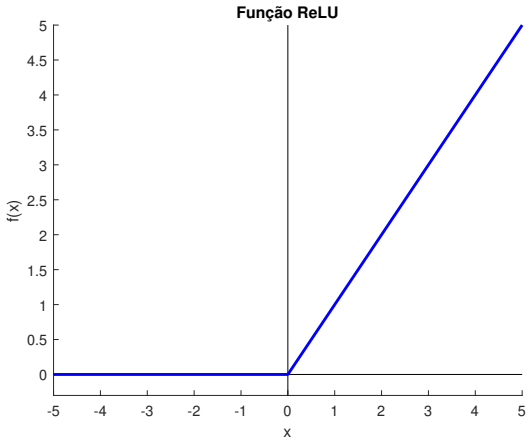

Figura 5 – Representação gráfica da função ReLU

Função Identidade

$$f(x) = x \tag{2.7}$$

Apesar de não contribuir para complexidade e capacidade de aprendizado da rede, esta função não limita os valores de sua saída a nenhuma faixa, podendo ser, em alguns casos, uma alternativa interessante para os neurônios da camada de saída.

A representação gráfica da função é vista na Figura 6.

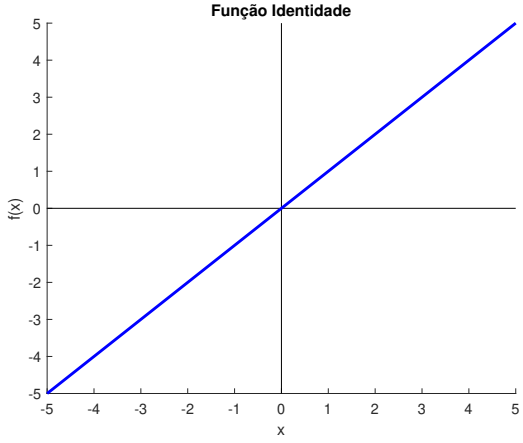

Figura 6 – Representação gráfica da função identidade.

2.3 *Backpropagation*

Para realizar o treinamento das redes, de acordo com [5], um algoritmo muito utilizado é o *backpropagation*. Nele, existe uma função custo, $C(W)$, que mensura o erro do conjunto de pesos, W, atual da rede. O negativo do gradiente desta função é utilizado, então, para calcular o ajuste que deve ser feito em cada um dos pesos de modo que o erro seja reduzido. Para o cálculo dos valores deste gradiente referentes às camadas anteriores a de saída, é necessário propagar este erro até cada uma das conexões por meio da regra da cadeia.

O cálculo da função de custo $C(W)$ referente ao conjunto atual de pesos W é definido pela Equação (2.8), onde p é o número de amostras do conjunto de treino utilizado; n é o número de saídas da rede; $y_i^{(j)}$ é o valor desejado na saída de índice i para a amostra j; e $\hat{y}_i^{(j)}$ é o valor estimado pela rede na saída de índice i para a amostra j.

$$C(W) = \frac{1}{p}\sum_{j=1}^{p}\sum_{i=1}^{n}(\hat{y}_i^{(j)} - y_i^{(j)})^2 \tag{2.8}$$

O cálculo do gradiente referente a cada uma das conexões da rede se dá através da Equação (2.9), onde $C(W)$ é a função custo calculada para os pesos atuais da rede; $y_i^{(L)}$ é

a saída do neurônio de índice i da camada L; $x_i^{(L)}$ é o somatório das entradas multiplicadas por seus respectivos pesos (ou seja, a saída antes de passar pela função de ativação) do neurônio de índice i da camada L; e $w_{ij}^{(L)}$ é o peso da conexão entre o neurônio de índice j da camada $L-1$ e o neurônio de índice i da camada L para o qual está sendo calculado o valor do gradiente.

$$\frac{\partial C(W)}{\partial w_{ij}^{(L)}} = \frac{\partial C(W)}{\partial y_i^{(L)}} \frac{\partial y_i^{(L)}}{\partial x_i^{(L)}} \frac{\partial x_i^{(L)}}{\partial w_{ij}^{(L)}} \tag{2.9}$$

Após calculado o valor do gradiente referente a cada uma das conexões da rede, o ajuste $\Delta w_{ij}^{(L)}$ a ser aplicado no peso da conexão $w_{ij}^{(L)}$ acontece conforme a Equação (2.10), onde η representa a taxa de aprendizado e $\frac{\partial C(W)}{\partial w_{ij}^{(L)}}$ é o valor do gradiente do erro para a conexão.

$$\Delta w_{ij}^{(L)} = -\eta \frac{\partial C(W)}{\partial w_{ij}^{(L)}} \tag{2.10}$$

Por fim, o ajuste é aplicado aos pesos da rede conforme a Equação (2.11).

$$w_{ij}^{(L)}{}_{atual} = w_{ij}^{(L)}{}_{anterior} + \Delta w_{ij}^{(L)} \tag{2.11}$$

2.4 Taxa de aprendizado

O valor da taxa de aprendizado tem influência direta no cálculo do ajuste dos pesos da rede. Ela define o quão grande será o passo na direção oposta a do gradiente calculado através do *Backpropagation*. A escolha de uma taxa de aprendizado muito elevada pode resultar em instabilidade no treinamento, onde o ajuste acaba ultrapassando a área ótima da solução e muitas vezes sendo incapaz de convergir para algum valor ótimo, seja global ou mesmo local. Por outro lado, a escolha de uma taxa muito pequena pode levar a um treinamento demasiadamente lento ou mesmo fazer com o treinamento fique preso em ótimos locais, não sendo capaz de levar a rede deste ponto para o ponto do ótimo global da solução. Exemplos gráficos de como diferentes valores de taxa de aprendizado se aproximam do ótimo da função são apresentados na Figura 7.

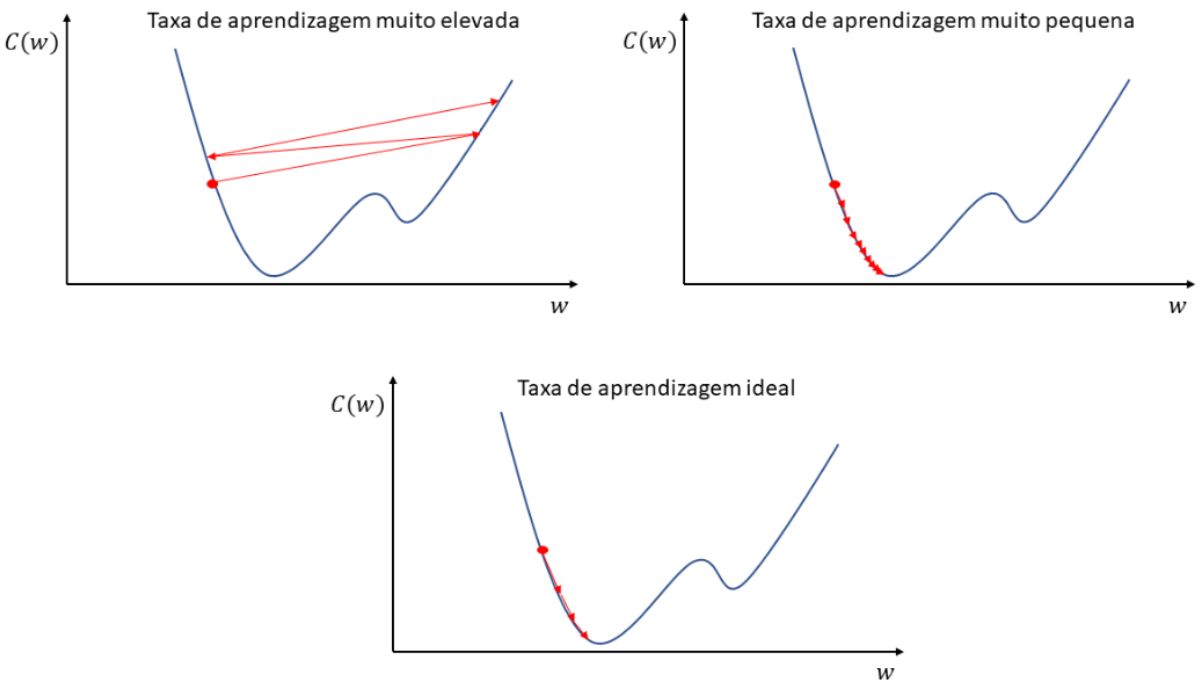

Figura 7 – Desempenho de diferentes taxas de aprendizado na convergência para o ótimo da função.

A taxa de aprendizado é, normalmente, um parâmetro livre da rede que não possui um valor ideal definido. Portanto, cabe ao usuário definir, de forma empírica, o valor da taxa a ser utilizada na rede específica. Como este é um parâmetro que possui forte influência nos resultados possíveis de se obter para uma rede, sua escolha pode ser trabalhosa e não intuitiva, sendo necessárias, muitas vezes, um grande investimento de tempo para encontrar o valor mais adequado para a taxa de aprendizado a ser utilizado na rede.

Para evitar a necessidade da realização desses testes até obter a taxa de aprendizado ideal para uma dada rede, foram desenvolvidos os algoritmos de taxa de aprendizado adaptativa. Com isso, embora ainda seja necessário definir uma taxa de aprendizado base para o método, este parâmetro passa a causar pouco impacto no treinamento da rede, pois o algoritmo modifica, a cada novo ajuste a ser aplicado aos pesos, a taxa de aprendizado para um valor ideal a ser utilizado naquele instante.

Apesar de tornar o processo de treinamento mais custoso computacionalmente, por adicionar uma etapa a mais ao cálculo dos ajustes, o uso de uma taxa de aprendizado adaptativa acaba por reduzir o tempo total gasto com o treinamento da rede, visto que elimina a etapa de realizar testes buscando a taxa ideal para, só depois de encontrada, ser aplicada no treinamento real da rede.

Alguns dos métodos propostos para cumprir esta função são capazes, ainda, de calcular uma taxa de aprendizado para cada um dos pesos da rede, não sendo necessário ajustar todos os pesos com um mesmo valor, como acontece no caso da taxa de aprendizado

fixa. Assim, é possível um treinamento muito mais eficiente, com uso de taxas maiores ou menores dependendo da necessidade, no momento, para cada um dos pesos da rede.

Alguns exemplos de métodos para cálculo de taxa de aprendizado adaptativa são o *Adagrad*, o *Adadelta* e o *Adam*. Estes três métodos funcionam de forma bastante semelhante, calculando uma taxa de aprendizado específica para cada conexão da rede, baseada no gradiente daquela conexão específica.

O método do *Adagrad*, visto em [7], calcula o ajuste dos pesos baseado numa taxa de aprendizado pré definida que é dividida pelo somatório dos quadrados dos gradientes anteriores, acumulados desde o início do treinamento da rede, de modo que o treinamento da rede é iniciado com uma taxa de aprendizado elevada, se aproximando de forma mais rápida da região ótima e reduzindo gradativamente a taxa para possibilitar uma acomodação suave neste ótimo. Porém, por acumular os gradientes desde o início do treinamento, eventualmente o denominador da divisão adquire um valor muito elevado, fazendo com que os ajustes aplicados tornem-se muito próximos a zero, impossibilitando o treinamento da rede além deste ponto.

Para resolver o problema da impossibilidade de treinar a rede além de certo ponto, o método do *Adadelta*, como visto em [8], transforma esse denominador não mais em um acumulo desde o início do treinamento, mas sim um acumulo apenas em uma janela de tempo. Assim, o método *Adadelta* funciona de forma muito semelhante ao *Adagrad*, porém levando em conta apenas os gradientes mais recentes, deixando de acumular valores indefinidamente e mantendo o denominador da equação num valor razoável, sendo possível a realização dos ajustes mesmo após muitas etapas de treinamento.

O método *Adam*, encontrado em [9], por sua vez, funciona de forma muito semelhante ao *Adadelta*, porém acrescentando, ainda, uma parcela a mais no cálculo dos ajustes que se assemelha com o uso de *Momentum*, muitas vezes utilizado junto ao método de ajuste padrão com taxa de aprendizado fixa. A aplicação de *Momentum* no treinamento é, basicamente, dividir o cálculo do ajuste dos pesos em duas partes: uma levando em conta a taxa de aprendizado e gradiente calculado no instante atual e a outra sendo uma parcela do ajuste realizado na iteração anterior do treinamento. A Equação (2.12) demonstra como é feito o cálculo do ajuste para um peso w no instante k utilizando *Momentum*, onde η é o valor da taxa de aprendizado, $g(w)_{(k)}$ é o gradiente do erro calculado para o peso w no instante k, $\Delta w_{(k-1)}$ é o ajuste aplicado ao peso w no instante $k-1$ e p é um valor entre 0 e 1, definido pelo usuário, que faz o balanço entre as duas partes da equação.

$$\Delta w_{(k)} = p \times (-\eta g(w)_{(k)}) + (1-p) * (\Delta w_{(k-1)}) \tag{2.12}$$

Cada um desses métodos possui ainda alguns parâmetros livres para definir seu funcionamento, como a taxa de aprendizado base, coeficientes de decaimento etc. Porém, apesar de aumentarem o número total de parâmetros livres a serem definidos na rede, estes parâmetros causam um impacto muito menor no treinamento da rede do que a taxa

de aprendizado em si, não sendo necessária uma escolha muito rigorosa para definição de seus valores.

2.5 Comentários Finais

Neste capítulo foram vistos conceitos básicos sobre o funcionamento das redes neurais e algumas opções de ferramentas para ajudar no treinamento das redes. No próximo capítulo, serão apresentados alguns conceitos e termos de identificação de sistemas que foram úteis para o desenvolvimento deste trabalho.

3 Identificação de Sistemas

Na tarefa de encontrar modelos matemáticos capazes de representar o comportamento de sistemas, sejam estes dinâmicos ou estacionários, foram desenvolvidas diversas técnicas e métodos ao longo do tempo [10]. Este conjunto de conhecimentos e ferramentas é o que define a área de identificação de sistemas.

Os modelos utilizados em grande parte das técnicas de identificação de sistema, como é o caso das redes neurais do tipo *feedforward*, são estáticos, não havendo relações dinâmicas entre suas entradas e saídas. Assim, para que seja possível representar as dinâmicas de um sistema através desses modelos, são adicionados regressores das entradas e saídas do sistema na composição de suas entradas. Estes regressores são as respectivas entradas e saídas do sistema com atrasos de amostra. Desta forma, o funcionamento do modelo se assemelha, neste aspecto, a maneira como são construídas as equações a diferenças.

Como exemplo, um sistema qualquer com entrada x e saída y, poderia ser representado por um modelo que leva em conta, para o cálculo da saída da amostra atual, para o instante k, $y(k)$, os regressores com atraso de 1 e 3 atrasos de amostra da entrada x e ainda os regressores com atraso de 1 e 2 amostras da saída y (considerando que estes seriam os regressores adequados para a representação deste sistema). Neste caso, as entradas do modelo seriam: $x(k-1)$, $x(k-3)$, $y(k-1)$ e $y(k-2)$.

De modo a mensurar a qualidade da representação do modelo, é realizada a validação do mesmo através de uma simulação livre em um conjunto de dados diferentes do utilizado para obtenção do modelo. Para realizar a simulação livre, é fornecido ao modelo as informações do estado inicial do sistema, considerando todos os regressores necessários para as entradas do modelo. Além disso, é fornecido o conjunto de entradas que alimentam o sistema desde este ponto inicial até o último ponto a ser estimado pelo modelo. Assim, o modelo estima as saídas do sistema para os instantes seguintes, passando a considerar, nas próximas iterações, as saídas estimadas pelo próprio modelo para compôr seu conjunto de regressores de entrada. Ou seja, com apenas o estado inicial do sistema e o conjunto de entradas aplicadas ao longo da simulação, estima-se seu comportamento para todo um conjunto de amostras.

A partir dos resultados desta simulação, as saídas estimadas pelo modelo são comparadas com as saídas conhecidas do sistema para este conjunto de dados. Esta diferença é o erro de estimação, existindo um valor de erro para cada uma das amostras. De modo a generalizar a qualidade da estimação do conjunto como um todo, são utilizados índices como o erro quadrático médio (MSE - do inglês, *Mean Squared Error*), R^2 etc.

Nas seções deste capítulo serão apresentados alguns conceitos de identificação de sistemas utilizados na obtenção de modelos baseados em dados.

3.1 *Overfitting*

O *overfitting* é um problema indesejado para um modelo, sendo caracterizado pela incapacidade do modelo de generalizar o comportamento de um sistema, representando este com qualidade apenas no conjunto de dados utilizados para o treinamento. O problema ocorre quando o modelo não aprende as relações de fato existentes entre as entradas e saídas do sistema, mas sim apenas decora os dados presentes no conjunto utilizado para treinamento.

A Figura 8 demonstra um exemplo do que seria o *overfitting*. Dado o sistema original, são coletados alguns dados que, por motivos diversos, tais como imprecisão dos sensores ou presença de ruído, não se encaixam perfeitamente na curva que representa o comportamento real do sistema. A partir destes dados, um bom modelo seria obtido de forma que represente razoavelmente bem todas as amostras e que tenha comportamento próximo ao do sistema real. No caso onde ocorre *overfitting*, o modelo é obtido de forma a representar perfeitamente os dados existentes no conjunto utilizado para identificação, porém representa um comportamento que nada tem a ver com o sistema real que está sendo identificado.

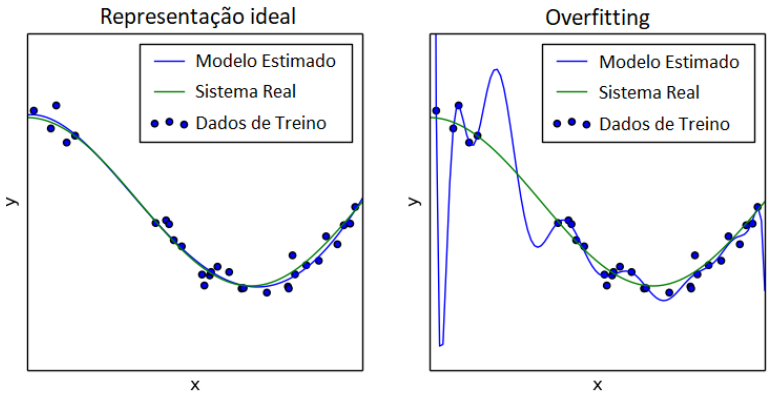

Figura 8 – Representações de um mesmo sistema, a partir de um modelo bem condicionado e outro modelo onde ocorre o *overfitting*.

Algumas das principais causas de *overfitting* são: conjunto de dados de treinamento com poucas amostras, complexidade excessiva do modelo utilizado para representar o sistema e, em casos de modelos neurais, o treinamento excessivo da rede. Todos as três causas levam ao mesmo problema: o modelo deixa de representar as relações entre as variáveis e suas dinâmicas e passa a simplesmente "decorar"o conjunto de dados. As Figuras 9, 10 e 11 demonstram, respectivamente, as relações entre tamanho do conjunto de dados, complexidade do modelo e excesso de treinamento de uma rede neural com o problema do *overfitting*.

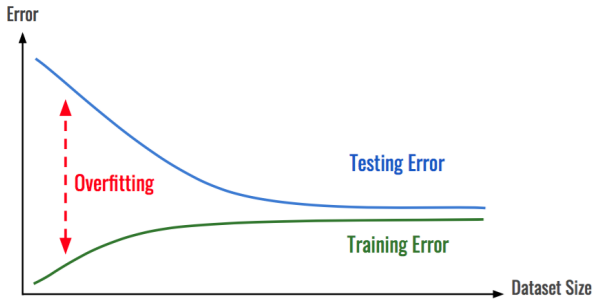

Figura 9 – Relação entre o tamanho do conjunto de dados e erro nos conjuntos de treino e de validação: Com um conjunto de dados demasiadamente pequeno, não é possível reconhecer todas as dinâmicas presentes no comportamento do sistema. Devido a isso, o modelo acaba por decorar as respostas do sistema para as poucas amostras de dados disponíveis, representando bem o comportamento do sistema para aquelas entradas específicas, mas sendo incapaz de generalizar este comportamento em outros conjuntos de dados.

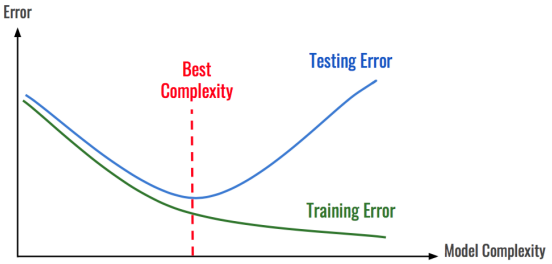

Figura 10 – Relação entre complexidade do modelo e erro nos conjuntos de treino e de validação: Com uma alta complexidade do modelo e consequente alto poder de processamento, o modelo tende a ter uma maior facilidade em simplesmente representar especificamente cada uma das amostras presentes no conjunto de dados de treinamento. Assim, melhora sua representação para estas amostras específicas, porém não adquire a capacidade de generalizar o comportamente do sistema em outros conjuntos de dados.

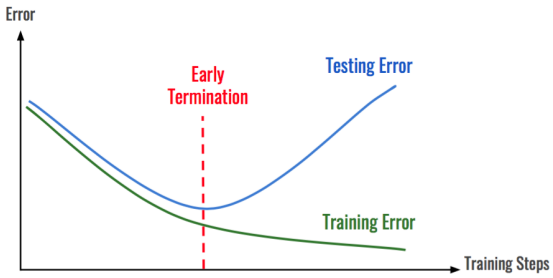

Figura 11 – Relação entre etapas de treinamento da rede realizadas e erro nos conjuntos de treino e de validação: O treinamento da rede acaba por forçá-la a representar o melhor possível aquele conjunto de dados utilizado no treinamento. Assim, quando feito em excesso, a rede é obrigada a especializar-se em representar muito bem as amostras contidas no conjunto de dados de treinamento. Com isso, acaba sacrificando sua capacidade de generalizar o comportamento do sistema e, por consequência, sua capacidade de representá-lo em deferentes conjuntos de dados.

3.2 Projeto de Testes

De acordo com [10], para que seja possível a obtenção de um modelo que seja capaz de representar as dinâmicas do sistema, é necessário que estas dinâmicas estejam presentes no conjunto de dados a partir dos quais será realizada a identificação. Portanto, é necessário uma taxa de amostragem adequada para a coleta dos dados, de modo que as dinâmicas não fiquem ocultas entre uma amostra e outra. Ainda, é necessário que os dados coletados não sejam de um momento de estado estacionário do sistema, onde as variáveis não mudam de valor e, por consequência, não é possível identificar suas dinâmicas.

Nos casos onde é possível excitar o sistema livremente para a coleta de dados, como em modelos virtuais ou plantas experimentais, é desejável um projeto de sinal de excitação de forma a abranger todas as faixas de operação desejadas do sistema e, em cada uma destas faixas, explorar variações com diferentes frequências. Para esta função, é comum o uso de sinais como *Pseudorandom Binary Sequence* (PRBS), *Pseudorandom Sequence* (PRS), combinação de senoides etc. Estes são boas opções por tratarem-se de sinais de excitação persistente com ricos espectros de frequência, capazes de extrair as diferentes dinâmicas do sistema em relação às suas diversas frequências.

Ainda, é necessário que sejam coletados dados em quantidade suficiente, pois um conjunto de dados muito reduzido pode não ser suficiente para a correta identificação do sistema, podendo levar o modelo a sofrer de *overfitting*.

3.3 Método de seleção de estrutura do FROLS - *Forward Regression OLS*

O estimador por mínimos quadrados ortogonais (OLS - *Orthogonal Least Squares*), encontrado em [11], é um método bastante consolidado na literatura e é utilizado para obter modelos não lineares. O método consta de realizar uma ortonormalização no conjunto de dados, tornando possível a estimação de cada parâmetro de forma individual. Ainda, é possível mapear o modelo ortonormal, obtido através da estimação, de volta para os domínios do sistema inicial.

A partir desta técnica, foi desenvolvido o método de estimação com seleção de estrutura FROLS, que usa como base para escolha dos parâmetros do sistema a figura de mérito da taxa de redução de erro em conjunto com a relação erro-sinal.

Taxa de Redução de Erro - ERR

Como visto em [10, 11], o modelo estimado do processo, por se tratar de uma aproximação, contém um erro residual em sua resposta. A resposta do sistema pode, então, ser representada pela soma de duas parcelas: a resposta do modelo estimado em função das entradas e o valor de erro residual. A Equação (3.1) demonstra essas parcelas, onde Y seria a resposta do sistema, x as entradas do modelo, $f(x)$ a função estimada para representar o modelo e e a parcela de erro residual.

$$Y = f(x) + e \tag{3.1}$$

Quanto mais uma entrada contribui para aproximar a resposta do modelo em função das entradas ao valor da resposta do sistema real, mais esta variável contribui para a redução do valor de erro residual. A taxa de redução de erro (ERR - *Error Reduction Ratio*) é o valor calculado para quantificar essa contribuição da entrada do modelo na redução do erro residual. Como visto em [10, 11], o valor da ERR para uma entrada de índice i do modelo é feito conforme a Equação (3.2), onde x_i é o vetor contendo as amostras da entrada de índice i do sistema e y é o vetor contendo as amostras de saídas do sistema. Este valor pode ainda ser multiplicado por 100 para indicar o valor percentual da contribuição da entrada.

$$ERR_i = \frac{<y, x_i>^2}{<x_i, x_i>} \tag{3.2}$$

As duas parcelas que compõem a resposta do sistema podem ser vistas, então, como o somatório dos valores de ERR das entradas presentes no modelo e a relação erro-sinal (ESR - *Error-to-Signal Ratio*). Sendo estas duas parcelas valores percentuais referentes a suas contribuições na composição da resposta do sistema. Portanto, a composição da

resposta do sistema é definida conforme a Equação (3.3), onde M é o número de entradas do modelo.

$$1 = \sum_{i=1}^{M} ERR_i + ESR \tag{3.3}$$

De forma equivalente, temos a Equação (3.4).

$$ESR = 1 - \sum_{i=1}^{M} ERR_i \tag{3.4}$$

Algoritmo da seleção de estrutura do FROLS

De acordo com [11], primeiro se calcula o valor de ERR para cada um dos regressores do modelo, selecionando o regressor com maior ERR.

Após selecionado este, o grupo de regressores candidatos é reduzido, com a retirada do regressor já selecionado. O conjunto de regressores remanescente é então ortogonalizado, através do método de *Gram-Schmidt*, em relação ao primeiro selecionado. A partir deste conjunto reduzido e ortogonalizado, é selecionado o regressor seguinte. O processo é repetido até que a condição de parada seja atingida.

A condição de parada acontece quando o valor do ESR, calculado a partir do somatório das ERR do conjunto de regressores selecionados até o momento, for menor que o valor ρ definido pelo usuário, como visto na Equação (3.5).

$$ESR \leqslant \rho \tag{3.5}$$

Após isso, o modelo é construído com os regressores selecionados, descartando os demais candidatos.

3.4 Comentários Finais

Neste capítulo, foram apresentados alguns conceitos de identificação de sistemas para obtenção e validação de modelos. No capítulo seguinte, serão apresentadas as técnicas e métodos que foram utilizadas no desenvolvimento deste projeto.

4 Metodologia

Neste capítulo serão apresentados como foram utilizados os conceitos postos nos capítulos anteriores para a resolução dos problemas propostos no Capítulo 1, assim como técnicas alternativas para a resolução de problemas que surgiram ao longo do desenvolvimento deste projeto.

Linguagem e bibliotecas

Levando em conta as necessidades do ambiente de trabalho, como ferramentas já existentes e planos futuros, foi escolhida a linguagem *Python* para implementação do projeto.

Para a implementação das redes neurais, foi feito uso da biblioteca *TensorFlow*, com a qual é possível, de maneira simples, montar uma estrutura de rede neural e realizar seu treinamento. Nessa biblioteca, já estavam disponíveis implementações de diversas funções de ativação e técnicas de taxa de aprendizado adaptativa. A função LReLU, assim como o método *Adam*, escolhidos para aplicação neste projeto, já estavam implementados na biblioteca.

A técnica de seleção de estrutura, porém, não possui implementação na biblioteca e, portanto, teve de ser desenvolvida por completo. Além da *TensorFlow*, foram utilizadas na implementação bibliotecas como *NumPy*, para lidar com dados e operações matemáticas/vetoriais, *time* para acompanhar o tempo levado em cada etapa do treinamento, seleção de estrutura etc. da rede e *pickle* para salvar dados, redes treinadas etc.

4.1 Função de Ativação

Inicialmente foram utilizadas funções de ativação do tipo ReLU nas camadas ocultas da rede. Porém, durante a execução dos treinamentos das redes, foram enfrentados problemas com a morte de neurônios e incapacidade da rede de convergir para uma boa representação do sistema nessas condições. De modo a resolver esse problema, foi adotado o uso da função *Leaky ReLU* (LReLU).

A função LReLU se comporta de maneira similar a original ReLU, porém para valores abaixo de zero, em vez de retornar zero, a função se comporta conforme a Equação (4.1), onde α é uma constante com valor $0 < \alpha < 1$.

$$f(x) = \alpha x \tag{4.1}$$

Desta maneira, é mantida a não linearidade da função, seu cálculo continua computacionalmente simples e o problema dos neurônios mortos é eliminado, visto que, com

o uso desta função, resultados abaixo de zero ainda possuem um gradiente e, portanto, ainda podem ser treinados em qualquer situação. Na Figura 12 é demonstrada a forma gráfica da função LReLU.

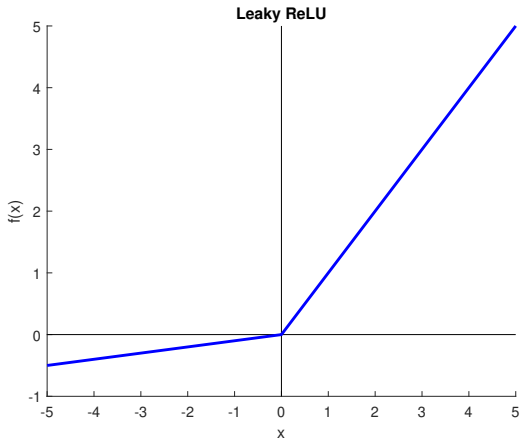

Figura 12 – Representação gráfica da função LReLU, com $\alpha = 0.1$.

4.2 Taxa de aprendizado adaptativa

De modo a reduzir o número de parâmetros livres da rede que causam grande impacto na qualidade do modelo obtido, foi adotado o uso de taxa de aprendizado adaptativa na rede.

O método escolhido foi o *Adam*, que calcula uma taxa de aprendizado adaptativa individual para cada conexão da rede. No cálculo do ajuste de peso, Δw, que se dá conforme a Equação (4.2), é utilizado um valor de taxa de aprendizado base, η, além de uma constante ϵ para evitar problemas numéricos e ajudar a manter a taxa de aprendizado bem condicionada mesmo em situações onde os gradientes estejam próximos de zero.

$$\Delta w_{(k)} = -\frac{\eta}{\sqrt{\hat{v}_{(k)} + \epsilon}} \times \hat{m}_{(k)} \tag{4.2}$$

As parcelas $\hat{m}$ e $\hat{v}$ são correções aplicadas as parcelas originais m e v, respectivamente. O cálculo das parcelas $\hat{m}$ e $\hat{v}$ se dão conforme as Equações 4.3 e 4.4, respectivamente. Nas equações, os valores β_1 e β_2 são as constantes de decaimento de m e v, respectivamente.

$$\hat{m}_{(k)} = \frac{m_{(k)}}{1 - \beta_1^k} \tag{4.3}$$

$$\hat{v}_{(k)} = \frac{v_{(k)}}{1 - \beta_2^k} \tag{4.4}$$

Estas correções são necessárias para adequar o valor das parcelas na Equação (4.2) no início do treinamento, visto que os valores de m e v, que representam, respectivamente, a

média decadente dos gradientes passados e a média decadente dos quadrados dos gradientes passados, são iniciados em zero.

Os valores de m e v são calculados conforme as Equações 4.5 e 4.6, respectivamente.

$$m_{(k)} = \beta_1 \times m_{(k-1)} + (1 - \beta_1) \times g_{(k)} \tag{4.5}$$

$$v_{(k)} = \beta_2 \times v_{(k-1)} + (1 - \beta_2) \times g^2_{(k)} \tag{4.6}$$

Uma vez calculado o ajuste Δw, a regra de aplicação deste à conexão é a mesma vista na Equação (2.11).

4.3 *Neural-FROLS*

Com o intuito de combater o problema do *overfitting* causado pela alta complexidade do modelo e buscando obter o modelo mais simples possível para representar, com qualidade, as dinâmicas do sistema, é proposta uma estratégia de seleção de estrutura para a rede.

A seleção consiste em avaliar todas as conexões existentes entre os neurônios da rede, definindo quantas e quais são as conexões relevantes para cada neurônio. É definido um limite mínimo de conexões relevantes para que o neurônio seja mantido e, caso o neurônio não atinja esse limite mínimo, o neurônio é excluído da rede. Esta avaliação ocorre em todos os neurônios, com exceção apenas dos neurônios de saída, visto que não seria interessante a exclusão de uma saída da rede. Ou seja, é possível a exclusão de qualquer neurônio presente nas camadas ocultas ou até mesmo de camadas inteiras da rede. Além disso, a analise é executada também na camada de entrada, existindo, então, a possibilidade de entradas da rede deixarem de existir.

A Figura 13 mostra os passos necessários para realização de um ciclo completo de treinamento e seleção de estrutura de uma rede. É realizado o treinamento da rede até que se possa considerá-la treinada. Após finalizado o treinamento, é feita uma análise de todas as conexões da rede (detalhada na Seção 4.3.1) e definido quais são os neurônios que devem permanecer na rede e quais devem ser retirados. Caso não haja nenhum neurônio a ser retirado, o ciclo é finalizado e a estrutura atual da rede é escolhida como estrutura resultante.

Caso haja neurônios a serem retirados da rede, é realizada a poda destes neurônios e a nova rede, gerada a partir da exclusão dos neurônios, com sua nova estrutura, é treinada. Finalizado o treinamento, a rede é testada num conjunto de dados de validação e tem seu desempenho comparado com o obtido pela rede antes da poda. Caso o desempenho da nova rede seja pior que o da anterior, a rede existente antes da poda é restaurada e o ciclo é finalizado, sendo esta rede restaurada o resultado final da seleção de estrutura.

No caso desta nova configuração desempenhar melhor no conjunto de dados de validação, é feita uma nova análise das conexões, de modo a verificar a possibilidade de reduzir ainda mais a estrutura. A partir daí, o ciclo se repete como explicado anteriormente.

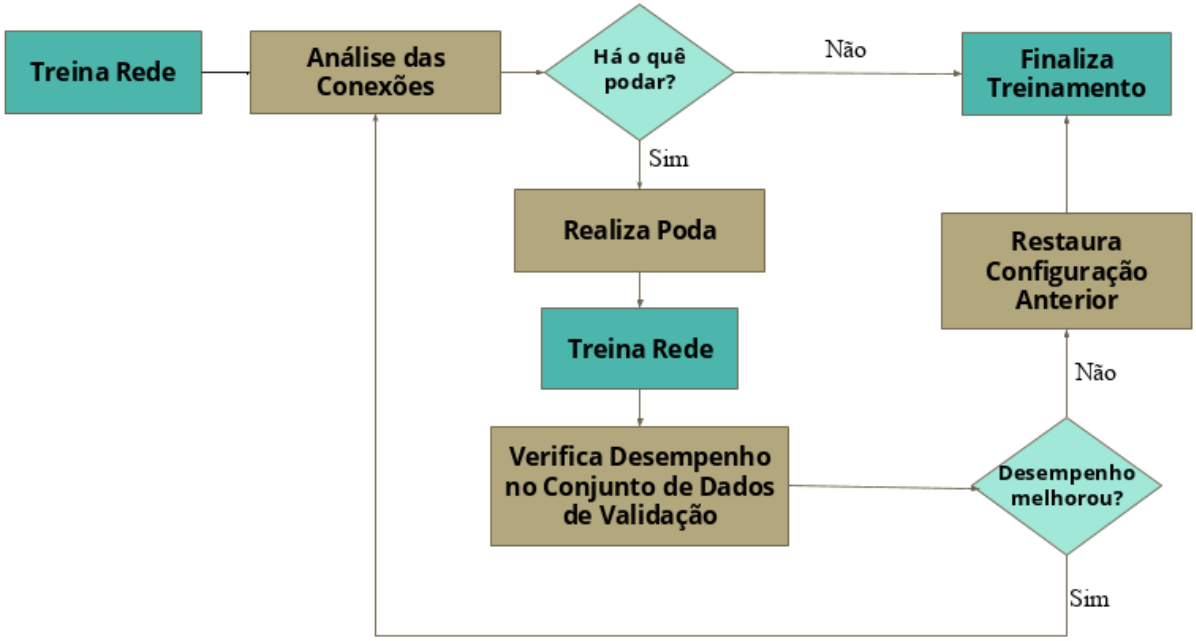

Figura 13 – Ciclo completo de treinamento e seleção de estrutura da rede

4.3.1 Análise das conexões e seleção dos neurônios

Para definir os neurônios a serem mantidos ou retirados da rede, é realizada, para cada um dos neurônios da rede, uma análise de suas conexões e um posterior cálculo de sua "pontuação". Após isso, apenas os neurônios que obtiveram uma pontuação igual ou acima do limite mínimo definido são escolhidos para permanecerem na rede. O método depende, ainda, de dois parâmetros livres: O valor mínimo de influência de uma conexão para que seja considerada relevante (ρ) e a pontuação mínima para que o neurônio seja mantido na rede ($lmin$). As etapas de análise das conexões e cálculo da pontuação, assim como a função dos parâmetros livres, são definidas a seguir.

Análise das conexões

A escolha das conexões se da por meio da análise, ao longo de um conjunto de dados conhecido, das entradas e das saídas de cada um dos neurônios. Assim como visto no algoritmo do FROLS, a análise é feita através da ortonormalização, através do método de *Gram-Schmidt* do conjunto de dados e cálculo do ERR. Para aplicar o método nas redes neurais, cada um dos neurônios é tratado como um modelo individual, conforme a Figura 14. Assim, cada neurônio é considerado como um modelo que tem como entradas

as saídas de todos os neurônios da camada anterior que se conectam a ele e como saída o valor do somatório do produto das entradas do neurônio por seus respectivos pesos. Ou seja, a saída do neurônio em si antes de ser aplicada a função de ativação deste.

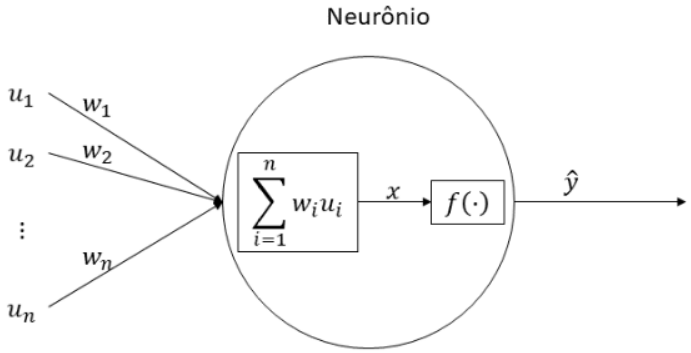

Figura 14 – Modelo do neurônio, onde u_i representa a entrada de índice i, w_i representa o peso da conexão de índice i, x é o valor do somatório do produto das entradas por seus pesos, $f(\cdot)$ é a função de ativação e $\hat{y}$ é a saída do neurônio.

O motivo para considerar a saída do modelo como sendo a saída do neurônio antes da aplicação da função de ativação é que existem casos onde a saída é a mesma para diferentes valores do somatório das entradas. No caso da ReLU, por exemplo, se o somatório das entradas resulta em qualquer valor entre 0 e $-inf$, a saída é sempre a mesma: zero. Assim, embora as entradas tenham suas contribuições individuais para a obtenção desse resultado, torna-se impossível quantificar a influência de cada uma das entradas na saída.

Assim como no método do FROLS, a cada iteração do algoritmo as conexões são analisadas e a que possuir maior relevância é selecionada. As conexões são selecionadas, uma a uma, até que o ESR seja igual ou menor que o valor ρ definido para o método. Assim, são formados os vetores contendo os índices de cada conexão relevante para cada um dos neurônios da rede.

Cálculo da pontuação do neurônio

A partir do vetor contendo as conexões de entrada que são relevantes para cada neurônio, obtido na etapa de análise das conexões, é gerado um novo vetor, contendo os índices dos neurônios da camada seguinte para os quais a saída do neurônio é considerada relevante. Por questões práticas, o primeiro vetor, obtido na etapa anterior, será chamado de $V_{entradas}$ e o segundo, obtido a partir deste, V_{saidas}.

Em posse dos dois vetores, $V_{entradas}$ e V_{saidas}, é realizado o cálculo da pontuação de cada neurônio, que definirá sua permanência ou exclusão da rede. O cálculo da pontuação tem como objetivo manter na rede apenas os neurônios que possuem múltiplas conexões com as camadas adjacentes. Deste modo, um neurônio n_r, que apenas repassa a informação

da camada anterior para a camada seguinte (ou seja, tem registrado apenas 1 conexão relevante em $V_{entradas}(n_r)$ e 1 conexão relevante em $V_{saidas}(n_r)$), assim como um neurônio n_i que não é considerado relevante para nenhum dos neurônios da camada seguinte (não possui nenhuma conexão registrada em $V_{saidas}(n_i)$) recebem pontuações baixas, enquanto neurônios que recebem múltiplas conexões relevantes e são considerados relevantes por múltiplos neurônios na camada seguinte recebem pontuações elevadas.

Para as camadas ocultas, o cálculo da pontuação é feito de acordo com a Equação (4.7), onde $Pontos(n)$ é a pontuação calculada para o neurônio n, $V_{entradas}(n)$ contém os índices das entradas consideradas relevantes para o neurônio n e $V_{saidas}(n)$ contém os índices dos neurônios para os quais a saída do neurônio n é considerada relevante.

$$Pontos(n) = (\text{Número de índices em } V_{entradas}(n)) \times (\text{Número de índices em } V_{saidas}(n)) \tag{4.7}$$

Após o cálculo, cada neurônio tem sua pontuação comparada ao limite mínimo definido através do parâmetro $lmin$ e, caso a pontuação seja maior ou igual a este limite, o neurônio permanece na rede, caso contrário, o neurônio é retirado.

Considerando um valor $lmin = 3$, podemos ver, na Figura 15, exemplos de neurônios que seriam excluídos (neurônios A, B, C, D e E) e neurônios que seriam mantidos (neurônios F, G e H) na rede. Neste caso, as pontuações dos neurônios A, B, C, D e E seriam, respectivamente, 1, 0, 2, 0 e 2. Como todas ficam abaixo do valor limite $lmin$, os neurônios seriam eliminados. Já os neurônios F, G e H, possuem, respectivamente, as pontuações 4, 3 e 3, ficando as três acima ou iguais ao limite mínimo $lmin$ e, portanto, sendo os neurônios mantidos na rede.

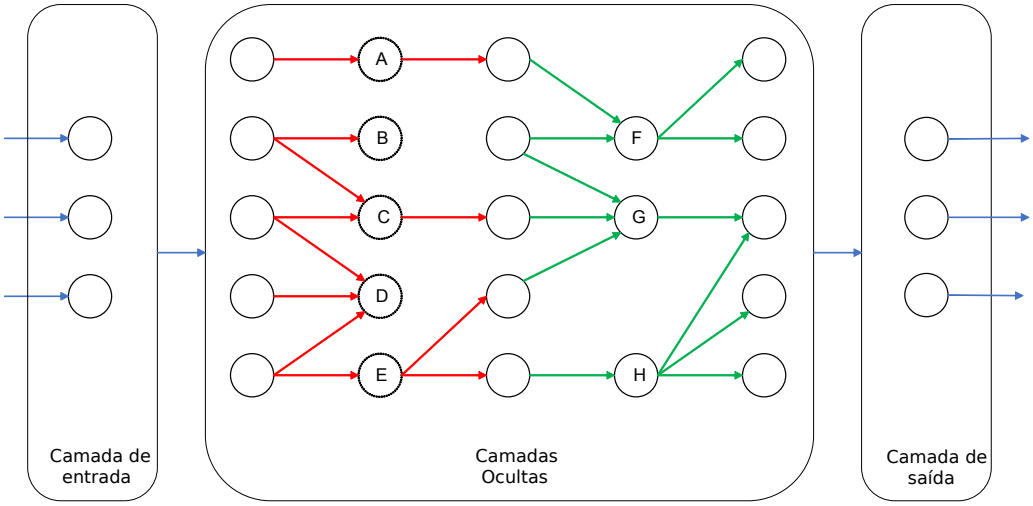

Figura 15 – Figura ilustrando neurônios com conexões insuficientes para serem mantidos na rede (neurônios A, B,C ,D e E) e neurônios com conexões suficientes para que permaneçam na rede (neurônios F, G e H).

Na camada de entrada, o cálculo das pontuações é feito de forma bastante similar ao caso das camadas ocultas. Nesta camada, como os neurônios recebem apenas os valores de entrada do conjunto de dados e não possuem nenhuma conexão vinda de outros neurônios, o valor do Total de indices em $V_{entradas}(n_{entrada})$, visto na Equação (4.7) é considerado 1 para qualquer neurônio $n_{entrada}$ da camada de entrada. Desta maneira, a pontuação dos neurônios da camada de entrada é calculada levando-se em conta apenas a quantidade de neurônios da camada seguinte para os quais o neurônio da camada de entrada é considerado relevante. Assim, a pontuação dos neurônios da camada de entrada é calculada conforme a Equação (4.8).

$$Pontos(n_{entrada}) = 1 \times (\text{Total de indices em } V_{saidas}(n_{entrada})) \tag{4.8}$$

Novamente considerando $lmin = 3$, a Figura 16 mostra exemplos de neurônios que seriam mantidos (B e C) e neurônios que seriam excluídos (A) da rede. No exemplo, o neurônio A tem a pontuação 2, abaixo de $lmin$ e portanto seria excluído, enquanto B e C teriam pontuações de valor 3, sendo, então, iguais ao limite mínimo estabelecido e, portanto, seriam mantidos na rede.

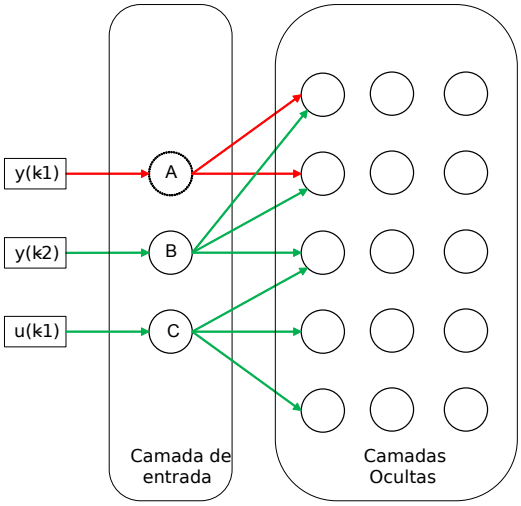

Figura 16 – Figura ilustrando entradas com conexões insuficientes para serem mantidos na rede (neurônio A) e entradas com conexões suficientes para que permaneçam na rede (neurônios B e C).

4.4 Comentários Finais

Neste capítulo foram vistas as técnicas utilizadas para resolver os problemas encontrados ao longo do desenvolvimento e alcançar os objetivos do trabalho. No próximo capítulo serão apresentados os resultados de um caso de estudo onde as técnicas foram aplicadas aos dados de um sistema de compressão de uma plataforma de petróleo e gás.

5 Estudo de caso: Sistema de compressão

Neste capítulo é apresentado o sistema de compressão de uma plataforma de petróleo utilizado como caso de estudo, seguido de detalhes sobre a implementação e os resultados da aplicação das técnicas apresentadas nos capítulos anteriores.

5.1 Funcionamento do Sistema

As unidades de produção de petróleo e gás natural são compostas basicamente por um separador trifásico, que recebe a produção dos poços, um sistema de tratamento de óleo/água e um sistema de compressão de gás, que é um processo vital da unidade. O sistema de compressão é responsável por aumentar a pressão do gás proveniente do separador, fornecendo uma determinada vazão de gás em pressão, umidade e temperatura específicas, de acordo com o ponto de operação desejado e as especificações dos sistemas subsequentes. O gás comprimido pode então ser exportado, usado para *gas-lift*, usado para gerar energia elétrica ou injetado novamente no reservatório nos poços de injeção [2, 3].

Figura 17 – FPSO (do inglês, *Floating Production Storage and Offloading*) utilizada na indústria de petróleo e gás

Os sistemas de compressão das unidades de produção de petróleo e gás são formados por compressores do tipo centrífugos, que trabalham em uma faixa relativamente estreita de condições operacionais. Em condições normais de operação, o sistema de compressão deve processar todo o gás recebido. Porém, alterações na vazão de gás proveniente do

separador ou nas propriedades do gás, como peso molecular e compressibilidade, podem ter impacto no funcionamento do compressor. Vários controladores locais são usados para manter pressões, temperaturas e vazão nos valores desejados para o funcionamento correto do compressor. No entanto, devido ao efeito de altas perturbações, mesmo com a ação do sistema de controle local, o sistema de compressão poderia ser levado a operar em regiões indesejáveis, tendo que queimar gás ou consumir mais energia do que o necessário. Em alguns casos extremos, o processo pode ficar instável, forçando o sistema de segurança a pará-lo. Uma parada do sistema de compressão afeta todos os outros sistemas da unidade, causando grandes perdas econômicas e estressando vários equipamentos [2, 3].

O sistema de compressão utilizado no caso de estudo deste trabalho conta com três compressores, o compressor principal, o compressor de exportação e o compressor de injeção. Um esquema simplificado do sistema de compressão é apresentado na Figura 18.

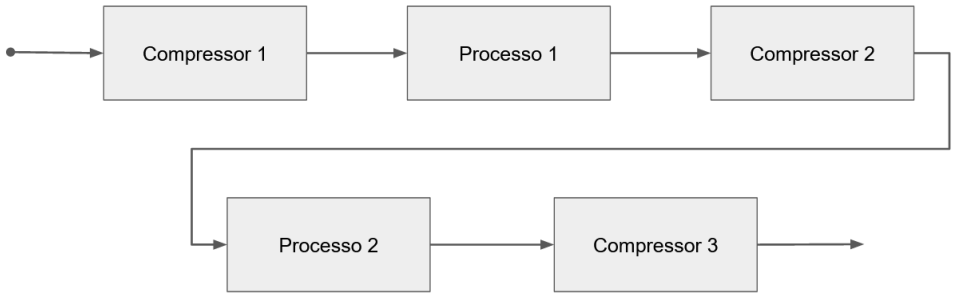

Figura 18 – Esquemático do sistema de compressão

O sistema de compressão trabalhado possui três compressores em série. Os Processos 1 e 2, apresentados na Figura 18, ainda não estão operando na planta do sistema e, portanto, não existe vazão de gás destes para fora do sistema. Por este motivo, é considerado que todo o gás que entra no sistema passa pelos 3 compressores antes de deixá-lo.

Cada compressor pode, ainda, contar com múltiplos estágios de compressão, podendo estes estarem dispostos em série ou em paralelo. Cada estágio de compressão conta com um vaso para receber o gás em sua entrada, o compressor em si, onde a pressão do gás é elevada, um trocador de calor para regular a temperatura do gás que é aquecido ao ser comprimido e uma válvula de reciclo.

A válvula de reciclo regula a passagem de gás da saída do estágio de compressão diretamente para o vaso existente em sua entrada. Em algumas circunstâncias o acionamento desta válvula é necessário para regular a diferença de pressão entre a entrada e a saída do compressor, evitando que este venha a operar em regiões instáveis do sistema.

Cada estágio conta ainda com dois controladores regulatórios. Um regula a pressão na entrada do compressor, atuando sobre a rotação do compressor, enquanto o outro mantém a relação de pressão entre a entrada e a saída do compressor em pontos de operação seguros, abrindo a válvula apenas em situações onde isso é necessário para

manter o sistema estável.

5.2 Variáveis de interesse

A Tabela 1 apresenta as variáveis utilizadas como entradas na modelagem do sistema. Já na Tabela 2, são apresentadas as variáveis de saída do sistema a serem representadas pelo modelo.

Entradas do Modelo	
Descrição	Quantidade
Setpoint dos controladores de pressão de sucção dos compressores (variáveis manipuladas)	3
Fluxo de entrada de gás no sistema (perturbação)	1
Total	**4**

Tabela 1 – Variáveis de entrada do sistema, manipuladas e perturbação.

Saídas do Modelo	
Descrição	Quantidade
Potência de operação nos estágios dos compressores	7
Medida de rotação dos trens de compressão	5
Pressões inter-estágio	3
Vazão de gás	2
Variáveis observadas no controle regulatório	17
Medidas alternativas da rotação dos compressores (redundantes)	2
Total	**36**

Tabela 2 – Variáveis de saída

5.3 Implementação

Aquisição de dados

Os dados utilizados para identificação do sistema foram obtidos através de simulações realizadas no software EMSO (sigla para *Environment for Modeling, Simulation and Optimization*). O software oferece um ambiente gráfico completo onde é possível modelar e simular complexos sistemas dinâmicos ou estacionários. Os componentes são modelados através de equações diferenciais que regem seu comportamento e modelos complexos podem ser construídos através de *flowsheets* onde são criados objetos destes componentes e são definidas todas as conexões existentes que formam o sistema como um todo.

A partir do *flowsheet*, onde o sistema de compressão está modelado, foram realizadas as simulações para aquisição de dados do sistema. Durante as simulações, foram aplicados, nas variáveis de interesse definidas como entradas, sinais compostos por somatórios de diversas senoides com diferentes amplitudes e frequências, de modo a levar o sistema para as diversas faixas de operação a serem analisadas e, nestas, variar as entradas para exibir as dinâmicas do sistema naquele ponto.

Rede Neural

A rede construída para identificar o sistema do caso de estudo foi do tipo MLP, possuindo 5 camadas, contendo, respectivamente, 1000, 100, 1000, 100 e 1000 neurônios cada uma. Foram selecionados os regressores da entrada com atrasos de 1 a 5, para todas as 4 entradas e 36 saídas da rede, somando um total de 200 regressores. Com isso, a rede foi montada com 200 neurônios na camada de entrada e 36 neurônios na camada de saída.

As funções de ativação empregadas foram do tipo LReLU para as camadas ocultas e função identidade para a camada de saída.

5.4 Resultados

Após realizado o treinamento, a rede representou de forma satisfatóra o sistema no conjunto de dados de treinamento. Porém, quando utilizada para simular o comportamento do sistema em um conjunto de dados de validação, diferente do utilizado no treinamento, a rede apresentou resultados ruins, demonstrando incapacidade de representar corretamente o comportamento do sistema. Os resultados da simulação e comparação entre os dados estimados e os produzidos pelo sistema real para uma das suas saídas podem ser conferidos na Figura 19.

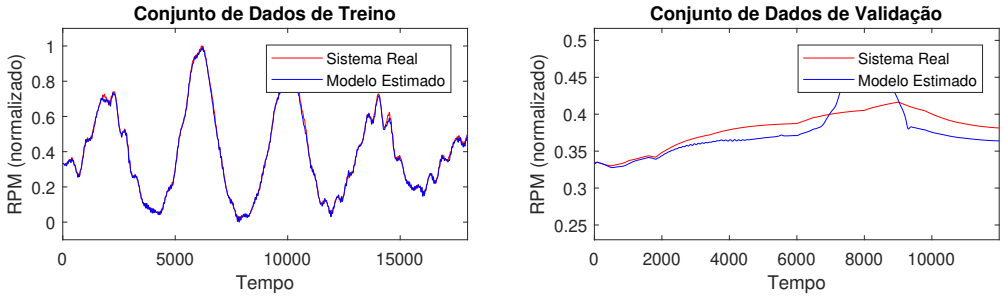

Figura 19 – Desempenho da rede em representar o sistema nos dados de treinamento e nos dados de validação.

Esta grande diferença de desempenho nos dois conjuntos de dados, obtendo uma representação quase perfeita no conjunto de dados de treinamento e um desempenho

insatisfatório no conjunto de dados de validação, indica a ocorrência de *overfitting* no modelo.

Visto que uma das principais causas de *overfitting* é o uso de um modelo excessivamente complexo para representar o sistema, foi executada a rotina de seleção de estrutura na rede. Os resultados da seleção de estrutura são apresentados na Tabela 3.

	Rede Original	Rede Após Seleção de Estrutura
Entradas da Rede (Total de Regressores)	200	139
Camadas Ocultas	5	5
Neurônios por Camada Oculta	[1000, 100, 1000, 100, 1000]	[180, 100, 159, 100, 16]
Saídas do Sistema que deixaram de ser entradas da Rede	-	2

Tabela 3 – Estrutura da rede antes e depois da seleção de estrutura

Após a seleção de estrutura, foi possível obter uma rede muito mais enxuta. Nesta nova rede obtida, alguns regressores deixaram de compor as entradas e houve uma grande redução na quantidade de neurônios totais da rede, chegando a uma redução de mais de 98% na quantidade de neurônios existentes na última camada oculta.

Ainda, podemos destacar que 2 saídas do sistema tiveram todos os seus regressores excluídos das entradas da rede. Este é um resultado bastante desejável, visto que estas duas saídas eram apenas variáveis utilizadas para monitorar o sistema, sendo seus dados redundantes.

Nos testes realizados utilizando esta nova rede, é perceptível uma queda na qualidade de representação do sistema no conjunto de dados de treinamento. Porém, no conjunto de dados de validação, o desempenho apresentado foi muito superior ao da rede original, demonstrando que a rede simplificada foi capaz de generalizar melhor o sistema, assim não apresentando o problema de *overfitting*. Os resultados das simulações da rede simplificada podem ser conferidos na Figura 20.

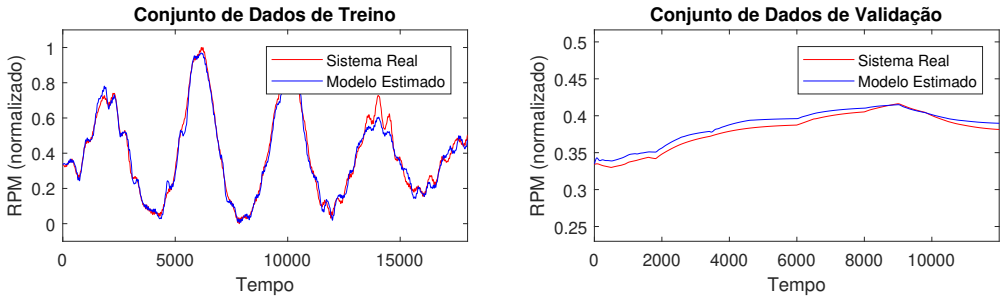

Figura 20 – Desempenho da rede em representar o sistema nos dados de treinamento e nos dados de validação após seleção de estrutura.

As Figuras 19 e 20 mostram resultados de estimações levando em conta apenas uma das 36 saídas presentes no modelo neural. Na Tabela 5 são apresentados, através do cálculo do MSE entre os dados de validação e os dados estimados pela rede, os resultados para as estimações de todas as saídas do modelo. As Figuras 19 e 20 mostram resultados referentes à saída de índice 11 vista na Tabela 5.

Ainda, a rede obtida após a seleção de estrutura apresentou uma redução de aproximadamente 15% no tempo de execução quando comparada com a rede original para um mesmo conjunto de dados. Já quando comparada com a simulação pelo modelo construído a partir de EDOs, através do software EMSO, a redução de tempo foi de cerca de 98% para os dados de validação e de cerca de 99,5% para o conjunto de dados de treinamento.

Os tempos de execução aproximados para os três casos, avaliados nos conjuntos de dados de treino e de validação, podem ser observados na Tabela 4.

	Rede Reduzida	**Rede Original**	**Simulador por EDOs**
Treino (18 mil amostras)	18 segundos	22 segundos	50 minutos
Validação (12 mil amostras)	12 segundos	14 segundos	10 minutos

Tabela 4 – Tempo de execução aproximado para estimação dos valores por cada um dos modelos, avaliados nos conjuntos de dados de treino e de validação.

O tempo de execução por amostra é constante para as redes neurais, visto que em toda execução são feitas as mesmas operações, apenas com valores diferentes. Já para o sistema baseado em EDOs, o ponto de operação em que o sistema se encontra e a ocorrência de determinados eventos podem influenciar fortemente o tempo de execução para cada amostra. Daí a justificativa de um aumento de 500% no tempo de execução para o modelo baseado em EDOs quando comparados os conjuntos de dados de validação e de treinamento, apesar de o conjunto de treinamento conter apenas 50% mais amostras que o de validação.

Índice da Saída da Rede	MSE das saídas da Rede Original	MSE das saídas da Rede Reduzida	Diferença (Original - Reduzida)
1	0,000039	0,000083	-0,000044
2	0,000143	0,000026	0,000117
3	0,000096	0,000092	0,000004
4	0,000201	0,000024	0,000177
5	0,000403	0,000048	0,000355
6	0,000554	0,000031	0,000523
7	0,003357	0,000042	0,003315
8	0,000611	0,000032	0,000579
9	0,044332	0,000157	0,044175
10	0,002175	0,000011	0,002164
11	0,000416	0,000053	0,000363
12	0,000567	0,000031	0,000536
13	0,003590	0,000040	0,003550
14	0,000758	0,000030	0,000728
15	0,008535	0,000099	0,008436
16	0,000059	0,000094	-0,000035
17	0.043298	0,000154	0,043144
18	0,000547	0,000028	0,000519
19	0,000029	0,000020	0,000009
20	0,000220	0,000028	0,000192
21	0,000091	0,000018	0,000073
22	0,000235	0,000031	0,000204
23	0,007232	0,000156	0,007076
24	0,001004	0,000035	0,000969
25	0,000732	0,000033	0,000699
26	0,000968	0,000027	0,000941
27	0,000270	0,000046	0,000224
28	0,000974	0,000034	0,000940
29	0,000580	0,000030	0,000550
30	0,000988	0,000027	0,000961
31	0,000303	0,000049	0,000254
32	0,002422	0,000016	0,002406
33	0,001705	0,000051	0,001654
34	0,000445	0,000142	0,000303
35	0,000790	0,000030	0,000760
36	0,000665	0,000029	0,000636
Soma Total:	0,129332	0,001878	0,127454

Tabela 5 – Tabela contendo os valores medidos de MSE no conjunto de dados de validação para a rede original e para a rede obtida após a seleção de estrutura.

5.5 Comentários Finais

Neste capítulo foram apresentados o sistema utilizado como caso de estudo, detalhes acerca da implementação das técnicas e aquisição de dados e ainda os resultados obtidos. No próximo capítulo serão apresentadas as conclusões e perspectivas futuras para este trabalho.

6 Conclusões

A aplicação das técnicas vistas no Capítulo 4 no treinamento da rede contribuiu com a redução dos parâmetros livres da rede. Com o uso da taxa de aprendizado adaptativa, foi possível treinar redes sem a necessidade de buscar um ajuste fino para a mesma.

A técnica de seleção de estrutura permitiu que bons resultados fossem obtidos mesmo com uma rede super-parametrizada e um escolha descuidada dos regressores a serem utilizados nas suas entradas. Isso possibilita que a rede seja construída sem muito critério quanto a sua complexidade, pois é possível simplesmente exagerar nos parâmetros da rede e, usando o seletor de estrutura, simplificar esta até que seja obtida uma complexidade ideal para representação do sistema.

Foi visto, ainda, que a técnica de seleção de estrutura apresentou bons resultados, eliminando entradas que deveriam, de fato, ser eliminadas do modelo, simplificando bastante a estrutura da rede e resolvendo o problema de *overfitting* existente na rede original. Com isso, a rede obtida após aplicação da técnica era muito mais qualificada para realizar estimações e simular o comportamento do sistema.

Ainda, durante os treinamentos foram enfrentados problemas com a morte de neurônios dvido ao uso da função de ativação ReLU. Com o uso da função alternativa *Leaky* ReLU o problema foi resolvido e a rede manteve um bom desempenho durante os treinamentos, com tempo de execução semelhante ao obtido com a função ReLU original.

Por fim, foi possível obter uma boa representação do sistema mesmo com a alta complexidade deste. Assim, a técnica mostrou-se promissora para aplicações em conjunto do MPC.

6.1 Trabalhos/Perspectivas Futuras

Apesar de a técnica de seleção de estrutura já apresentar bons resultados, algumas melhorias já foram planejadas para estudos, implementações e testes futuros. Dentre elas, a redução dos parâmetros livres da técnica, tornando ρ um parâmetro definido através de algoritmo, de maneira semelhante ao que é feito no caso das taxas de aprendizado adaptativas.

Outra melhoria para o método de seleção de estrutura seria encontrar uma função capaz de definir de maneira clara o *trade-off* entre representabilidade do sistema e complexidade da rede. Esta mudança permitiria uma seleção de estrutura com foco na obtenção do modelo com melhor representação possível ou, alternativamente, a busca por um modelo que sacrifique um pouco sua qualidade de representação para obter uma estrutura ainda mais simplificada.

Ainda sobre o seletor de estrutura, pode-se estudar a possibilidade de, além de

retirar neurônios por completo da rede, permitir que o algoritmo exclua também conexões específicas dos neurônios remanescentes.

Por fim, considerando o tempo de simulação aproximado de 1 milissegundo por amostra, as redes neurais se mostram uma opção viável para se utilizar em conjunto de técnicas como o PNMPC. Assim, a parcela de tempo necessária para a simulação do modelo não linear, que normalmente não é maior que 500 amostras, seria pequena em relação ao tempo de iteração do laço de controle, que normalmente tem valores como 5, 10 ou até mais segundos.

Referências

1 PLUCENIO, A. et al. A practical approach to predictive control for nonlinear processes. *IFAC Proceedings Volumes*, Elsevier, v. 40, n. 12, p. 210–215, 2007. Citado na página 13.

2 PLUCENIO, A. et al. Mpc advanced control of an offshore gas compression system. In: OFFSHORE TECHNOLOGY CONFERENCE. *OTC Brasil*. [S.l.], 2017. Citado 3 vezes nas páginas 13, 41 e 42.

3 VETTORAZZO, C. M. *Model Predictive Control of Gas Compression Station in Off-shore Production Platforms*. 122 p., 2016. Citado 3 vezes nas páginas 13, 41 e 42.

4 MONTáNS, F. J. et al. Data-driven modeling and learning in science and engineering. *Comptes Rendus Mécanique*, v. 347, n. 11, p. 845 – 855, 2019. ISSN 1631-0721. Data-Based Engineering Science and Technology. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1631072119301809>. Citado na página 13.

5 NELLES, O. *Nonlinear system identification: from classical approaches to neural networks and fuzzy models*. [S.l.]: Springer Science & Business Media, 2013. Citado 3 vezes nas páginas 17, 18 e 22.

6 HAYKIN, S. *Redes neurais: princípios e prática*. [S.l.]: Bookman Editora, 2007. Citado na página 17.

7 DUCHI, J.; HAZAN, E.; SINGER, Y. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, v. 12, n. Jul, p. 2121–2159, 2011. Citado na página 25.

8 ZEILER, M. D. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012. Citado na página 25.

9 KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. Citado na página 25.

10 AGUIRRE, L. A. *Introdução à identificação de sistemas–Técnicas lineares e não-lineares aplicadas a sistemas reais*. [S.l.]: Editora UFMG, 2014. Citado 3 vezes nas páginas 27, 30 e 31.

11 BILLINGS, S. A. *Nonlinear system identification: NARMAX methods in the time, frequency, and spatio-temporal domains*. [S.l.]: John Wiley & Sons, 2013. Citado 2 vezes nas páginas 31 e 32.