Séries temporais com MRFO e RNN

Fabrício Steinle Amoroso, Weslley Santana Barreto Rosalem
Departamento de Computação
Universidade Estadual Paulista "Júlio de Mesquita Filho"
Bauru, Brasil
fabricio.amoroso@unesp.br, w.rosalem@unesp.br

***Resumo*—Otimização de forrageamento de arraia manta (MRFO) é uma meta-heurística inspirada no comportamento adotado por arraias manta durante seu forrageamento. O algoritmo baseia-se em três técnicas de alimentação: forrageamento em corrente, em ciclone e através de cambalhotas. Neste artigo é apresentada uma aplicação do algoritmo MRFO para um problema de previsão utilizando dados de séries temporais. Foi possível observar que o uso do MRFO para ajustar os hiperparâmetros da arquitetura RNN para previsão de séries temporais, demonstrou uma melhoria significativa no desempenho.**

I. Introdução

De forma a otimizar o desempenho de modelos de aprendizado de máquina ao trabalhar com dados temporais, é proposta a utilização de RNNs juntamente ao algoritmo meta-heurístico MRFO.

Redes neurais recorrentes possuem características que podem ser bem aproveitadas ao trabalhar com séries temporais. Neste tipo de modelo supervisionado os nós são capazes de armazenar informações para serem retroalimentadas em iterações futuras. Desta forma é possível levar em consideração dados anteriores no processamento atual.

Em conjunto a RNN, utilizar o algoritmo de otimização MRFO, baseado nos comportamentos eficazes de alimentação das arraias manta, pode trazer bons resultados no problema proposto: previsão de picos de uso de memória de um servidor através de dados de série temporal.

II. Fundamentação Teórica

A. Manta Ray Foraging Optmization

O algoritmo MRFO, baseado em inteligência de enxame, tem como inspiração as arraias manta, uma das maiores criaturas marinhas conhecida. Podem ser divididas entre duas espécies: arraia de recife (manta alfredi), que chega até aproixmados 5.5 metros largura e a arraia gigante (manta birostris), que alcança 7 metros de largura.

Arraias são animais filtradores, possuindo além da sua boca filtradora, lóbulos cefálicos utilizados para canalizar água e alimento para dentro de sua boca. Suas principais fontes de alimento são zooplâncton e krill, que podem ser encontrados espalhados pelos oceanos. Além disso, uma arraia adulta pode chegar a consumir aproximadamente 5 quilos de alimento diariamente.

As arraias são seres sociais e costumam viajar em grupos de até 50 indivíduos. De forma a otimizar sua ingestão de alimentos as arraias possuem algumas técnicas que se beneficiam de sua organização em grupos.

A partir dos hábitos de alimentação das arraias, o algoritmo MRFO é composto pela modelagem matemática de três técnicas para obter otimização nos resultados: forrageamento em corrente, em ciclone e em cambalhotas [1].

- Forrageamento em Corrente: Forrageamento em grupo, no qual as arraias se alinham, formando uma corrente. Desta forma, conseguem fazer uma varredura, afunilando o alimento, diminuindo a quantidade de zooplâncton ou krill deixada para trás.
- Forrageamento em Ciclone: Assim como no método anterior, é formada uma corrente, nesta técnica, quando o grupo de arraias encontra uma área de alta concentração de alimento, se organizam de forma e nadar em círculos, em direção ao centro, formando uma espiral, como um "ciclone", que faz com que o alimento seja agrupado e levado em direção à superfície, fazendo com que se torne fácil para as arraias acessarem o alimento.

- Forrageamento em Cambalhotas: Esta técnica geralmente realizada em áreas com grande concentração de alimento, baseia-se em movimentos, como cambalhotas realizados pelas arraias, permitindo que os indivíduos adentrem a área com bastante alimento e se mantenham dentro dela ao realizar cambalhotas, ingerindo alimento ao longo de todo o movimento. Além disso, o movimento circular faz com que o alimento seja levado pela corrente de água criada até a boca das arraias.

Em termos de algoritmo, cada técnica é traduzida em um comportamento a ser seguido para as partículas:

- Forrageamento em Corrente: Quanto maior a concentração de alimento, melhor é considerada a posição, mesmo que a solução ótima ainda não seja conhecida. Tendo isso em vista, no forrageamento as partículas se movem em direção ao ponto de maior concentração. Além disso, seguindo a ideia da corrente, cada partícula se moverá também em direção a partícula (arraia) à sua frente, exceto pela primeira, que se moverá apenas em direção ao melhor ponto conhecido.
- Forrageamento em Ciclone: Similar ao forrageamento em corrente, cada arraia se move em direção ao indivíduo à sua frente e também ao melhor ponto conhecido, entretanto no forrageamento em ciclone esse movimento será feito em espiral. Seguir o melhor ponto conhecido aprimora a capacidade de explotação do algoritmo. Alternativamente, ainda dentro da técnica do ciclone, as arraias podem seguir pontos aleatórios ao invés de seguir a melhor posição conhecida, possibilitando também uma busca exploratória.
- Forrageamento em Cambalhotas: Para esta técnica, a posição do alimento é um pivô para realizar cambalhotas, de forma a se movimentar e buscar a melhor posição dentro de um raio da melhor área conhecida. Esta movimentação contribui para a explotação.

Quando utilizadas em conjunto, as três técnicas descritas têm o potencial de encontrar ótimos globais, tendo em vista que cada uma trabalha um aspecto diferente da busca: exploração e explotação [1].

B. Recurrent Neural Networks

Rede neural recorrente (RNN) é um modelo de aprendizado de máquina supervisionado que funciona como uma rede neural artificial (ANN), mas possui neurônios nas camadas ocultas capazes de entrar em laços de retroalimentação e armazenar informações entre iterações [2].

Devido a suas características, RNNs se tornam muito viáveis para trabalhar com problemas que utilizam de séries temporais, podendo levar em conta a entrada de iterações anteriores.

III. METODOLOGIA

A. Conjunto de Dados

O *dataset* utilizado neste artigo, contém dados históricos de um servidor web, em que métricas de utilização e consumo de recursos como mémoria, cpu, disco e rede foram coletadas em tempo real e armazenados em formato de série temporal.

Nos ambientes modernos de tecnologia da informação, a avaliação rigorosa do desempenho e da confiabilidade de sistemas e aplicações é um pré-requisito essencial para garantir a satisfação do usuário e a continuidade dos negócios. Entre as várias ferramentas e técnicas disponíveis para o monitoramento de sistemas, o uso de séries temporais para analisar métricas tem sido um dos métodos mais eficazes.

Séries temporais referem-se a conjuntos de dados coletados em intervalos sequenciais de tempo, onde cada ponto de dados é associado a data e hora. Este tipo de dados é essencial para analisar tendências, detectar anomalias e fazer previsões, especialmente em ambientes onde os estados dos sistemas podem variar dinamicamente.

B. Plataforma Utilizada

A plataforma de código aberta escolhida, para coletar e armazenar as métricas foi o Prometheus, por ser facilmente escalável e capaz de colectar milhares de métricas de diversos dispositivos em tempo real. Ele foi projetado principalmente para confiabilidade e desempenho, permitindo a coleta de métricas de sistemas e aplicações com alta eficiência e precisão, além fazer parte do ecossistema da *Cloud Native Computing Foundation* (CNCF) e tem sido amplamente adotado em ambientes de microsserviços e Kubernetes.

C. Métricas

Root Mean Squared Error (RMSE) e *Mean Absolute Error* (MAE) foram as métricas utilizadas para avaliar quão precisas foram as previsões do modelo em comparação aos dados reais coletados pelo Prometheus.

São métricas comumente aplicadas para avaliar o desempenho de modelos de aprendizado de máquina em tarefas de regressão, incluindo modelos preditivos em séries temporais.

RMSE é a raiz quadrada da média dos quadrados das diferenças entre os valores previstos pelo modelo e os valores reais. Ele serve para quantificar o quão distante, em média, as previsões do modelo estão dos valores reais. A fórmula para calcular o RMSE é:

$$RMSE = sqrt(\Sigma(P_i - O_i)^2/n)$$

Onde:

P_i é o valor previsto pelo modelo para a i-ésima observação.
O_i é o valor real da i-ésima observação. n é o número total de observações. sqrt representa a raiz quadrada. O RMSE penaliza mais os erros grandes, pois os quadrados das diferenças fazem com que erros maiores tenham um impacto proporcionalmente maior no resultado final.

MAE (*Mean Absolute Error*): MAE é a média das diferenças absolutas entre os valores previstos e os valores reais. Ele também serve para quantificar o quão distante, em média, as previsões do modelo estão dos valores reais, mas de maneira menos sensível a erros grandes em comparação ao RMSE. A fórmula para calcular o MAE é:

$$MAE = \Sigma|P_i - O_i|/n)$$

Em que:

P_i é o valor previsto pelo modelo para a i-ésima observação.
O_i é o valor real da i-ésima observação. n é o número total de observações.

Pela caracteristica da natureza do funcionamento de um servidor *web*, objeto de pesquisa deste experimento, foi utilizada a métrica ***"node_memory_Active_bytes"***, como seu próprio nome sugere, ela é responsável por armazenar consumo de mémoria em tempo real do dispositivo alvo.

Para avaliação de desempenho do modelo, dado o caso de uso e caracteristicas do problema, a métrica erro médio absoluto (MAE) foi considerada como a melhor forma de avaliação. Para tal, os parâmetros da função de perda da arquitetura *Long Short-Term Memory* (LSTM) e uma arquitetura de RNN, adequada para processar sequências de dados, foram adaptados considerando o MAE, já que ela penaliza o modelo pela diferença absoluta entre os valores previstos e reais, o que encoraja o modelo a prever o valor exato da série temporal. Ao contrário da RMSE que penaliza o modelo por grandes erros e não encoraja diretamente o modelo a prever o valor exato da serie temporal.

D. Experimentos

O experimento realizado consiste em comparar os resultados obtidos entre implementações de RNN no problema de séries temporais proposto com e sem a aplicação do MRFO.

IV. Resultados e Discussões

Abaixo é possivel visualizar o resultado obtido através da arquitetura RNN sem utilizanção do *Manta Ray Foraging Optimization, MRFO*, na otimização de parametros do modelo. Neste caso os parâmetros utilizados foram:

- *n_steps* = 3
- *activation= relu*
- *n_features* = 1
- *optimizer = adam*
- *loss = mae*

Através destes parametros, sem busca por otimização, foram realizadas 20 execuções e calculado o número médio das métricas *RMSE* e *MAE*. Que alcançaram os resultados abaixo:

- *Mean RMSE*: 80924639.87999392
- *Mean MAE*: 69163153.79070708

Já com a execução utilizando o algoritmo MRFO (Manta Ray Foraging Optimization) para otimizar os hiperparâmetros da arquitetura de rede neural recorrente (RNN) com uma camada LSTM (Long Short-Term Memory) para previsão de séries temporais. Houve uma melhora significativa no desempenho do modelo, sendo que o espaço de busca se concentrou apenas na otimização dos parametros abaixo:

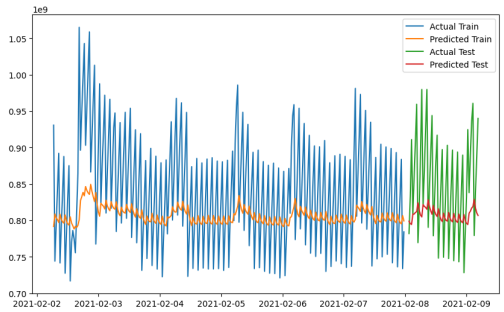

Figura 1. Figura criada pelos autores

- *n_units*: Número de unidades (neurônios) na camada LSTM. Isso afeta a capacidade do modelo em capturar padrões complexos na série temporal.
- *n_epochs*: Número de épocas, ou seja, quantas vezes o modelo é treinado em todo o conjunto de treinamento. Isso afeta a convergência do modelo durante o treinamento.

Os resultados obtidos com a utilização destes parâmetros otimizados foram satisfatórios, com:

- Mean RMSE: 67892782.18237053
- Mean MAE: 58680481.61098989

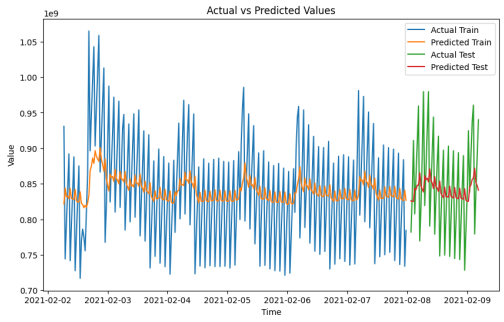

Figura 2. Figura criada pelos autores

V. Considerações Finais

O uso do algoritmo de otimização *MRFO* para ajustar os hiperparâmetros da arquitetura de rede neural recorrente com uma camada LSTM (Long Short-Term Memory) para previsão de séries temporais, demonstrou uma melhoria significativa no desempenho da rede, conforme evidenciado pela redução substancial nas métricas de erro, RMSE (Root Mean Square Error) e MAE (Mean Absolute Error).

O emprego do algoritmo MRFO permitiu uma busca eficaz no espaço de hiperparâmetros, o que culminou na identificação de um conjunto de hiperparâmetros que otimizaram o ajuste do modelo aos dados. Notavelmente, o RMSE e o MAE, ambos indicadores de erro de previsão, foram drasticamente reduzidos em comparação com a execução da rede neural sem otimização de hiperparâmetros.

No entanto, é fundamental reconhecer que, embora a otimização de hiperparâmetros possa melhorar o desempenho, ela não resolve todos os problemas de otimização e adaptação de uma rede neural. A qualidade dos dados de entrada e a seleção de características relevantes continuam sendo aspectos centrais para o sucesso de qualquer modelo de aprendizado de máquina. Além disso, é importante estar atento ao risco de *overfitting*, garantindo que o modelo não apenas se ajuste bem aos dados de treinamento, mas também generalize bem para dados não vistos.

Para os trabalhos futuros, será considerado a expansão do conjunto de features de entrada, não restringindo apenas as métricas de consumo de memoria, mas também de consumo de CPU, velocidade de escrita e leitura em disco, throuput de rede e quantidade de processos são algumas das features adicionais que serão consideradas . Elas permitirão que o modelo capture relações mais complexas e ofereça insights mais profundos sobre o problema em questão. Esse enriquecimento de dados de entrada pode ser essencial para descobrir padrões ocultos e melhorar ainda mais a precisão das previsões. Além disso, continuará sendo essencial explorar outras técnicas de otimização, incorporar regularização para mitigar overfitting, e avaliar o desempenho do modelo em outros conjuntos de dados.

Referências

[1] W. Zhao, Z. Zhang, and L. Wang, "Manta ray foraging optimization: An effective bio-inspired optimizer for engineering applications," *Engineering Applications of Artificial Intelligence*, vol. 87, p. 103300, 2020.

[2] H. Salehinejad, S. Sankar, J. Barfett, E. Colak, and S. Valaee, "Recent advances in recurrent neural networks," *arXiv preprint arXiv:1801.01078*, 2017.