Implementação e Avaliação de Redes Neurais Recorrentes com Uso de Técnicas de Espaço de Fase e Embedding na Modelagem de Séries Temporais

Igor Mol

igor.mol@makes.ai

6th November 2023

1 Long Short-Term Memory

O código–fonte contido no arquivo `LSTM.py` materializa a implementação de um modelo de redes neurais Long Short-Term Memory (LSTM) designado para a modelagem de séries temporais. A primeira etapa engloba os procedimentos de pré-processamento dos dados, que abarcam a normalização e a subsequente constituição dos conjuntos destinados ao treinamento e teste. Conduz-se então o aprendizado do modelo, a geração de previsões e, por fim, a elaboração de um gráfico capaz de demonstrar o confronto visual entre o modelo e os dados originais.

2 Redes Neurais Recorrentes

O arquivo denominado `recurrent_neural_networks.py` implementa uma abordagem de redes neurais aplicadas à modelagem de uma série temporal por meio de *embedding*, utilizando o modelo Recurrent Neural Networks (RNN). Inicialmente, os dados são normalizados e preparados para o treinamento. Posteriormente, cria-se e treina-se a rede neural RNN, realizam-se previsões e calcula-se o erro quadrático médio. Finalmente, apresentam-se os resultados, confrontando as previsões do modelo com os dados originais e ilustrando a tendência do erro ao longo do tempo.

3 Método de Espaço de Fase

O programa denominado `state_space.py` põe em prática a modelagem de uma série temporal utilizando um método de espaço de fase, efetuando a divisão dos dados em conjuntos de treinamento e teste, ajustando o modelo com base nos dados de treino e produzindo previsões para os dados

de teste. Para cada tamanho de conjunto de teste distinto, calcula-se e ilustra-se o erro quadrático médio entre as previsões e os respectivos dados de teste. Os dados originais, as previsões realizadas e o erro quadrático médio são todos representados em um gráfico.

4 Aprendizagem Supervisionada e *Embedding* de Takes

O código–fonte entitulado `takens.py` implementa a técnica de embedding de Takens com a finalidade de modelar uma série temporal. O método cria uma série "atrasada" com um número específico de dimensões, a ser determinada *a priori*, cujas especificidades podem ser ajustadas de acordo com a natureza e a magnitude dos dados que descrevem a série temporal. Posteriormente, calcula-se o erro quadrático médio entre as previsões feitas por um modelo *k–nearest neighbors* (KNN), que é treinado para modelar a série temporal.

5 Auto-Regressive Integrated Moving Average

O arquivo `ARIMA.py` contém a implementação do método "Auto-Regressive Integrated Moving Average" (ARIMA) com o propósito de modelar séries temporais. A versão particular do modelo ARIMA que usamos é utilizada para processar os dados de treinamento, prognosticando assim os n–passos subsequentes da série temporal estabelecida, onde $n := 50$ por padão. Notemos que este valor pode ser alterado de acordo com a necessidade imposta pelos dados que constitutem a série temporal a ser analisada. O resultado, por fim, é impresso em um diagrama, o qual exibe tanto os dados iniciais quanto o modelo devidamente ajustado, juntamente com as previsões referentes aos passos futuros.

O código-fonte `ARIMA_sliding_window.py` implementa o método ARIMA seguindo uma abordagem de janelas deslizantes. Duas listas são iniciadas pelo código: `history`, que contém os primeiros elementos da série temporal determinados pelo tamanho da janela estabelecida, e `predictions`, que armazena as previsões para etapas futuras. Posteriormente, itera-se ao longo da série temporal, construindo e ajustandoum modelo ARIMA com base na lista `history`, além de realizar uma previsão em n–passos, onde n se refere ao `horizonte`. Em conclusão, acrescentamos os dados reais após todas as previsões terem sido realizadas, e calculamos o erro quadrático médio, e encerramos a execução com uma comparação entre os dados reais e predições mediante a representação gráfica.

6 Redes Neurais Convolucionais

O programa cujo código-fonte está contido no script `cnn.py` cria um modelo de aprendizado de máquina para prever séries temporais usando Redes Neurais Convolucionais (CNN). Primeiramente, define-se a janela de dados e o horizonte de previsão, e transforma os dados em uma *sequência* de janelas. A parte central do código é a construção e o treinamento do modelo CNN, usando a

biblioteca Keras. Posteriormente ao treinamento, o modelo é usado para fazer previsões nos dados de treinamento. Adicionalmente, o código faz uma análise visual dos resultados, comparando a previsão com os dados originais em um gráfico. Ao final, o Erro Quadrático Médio (MSE), uma métrica de avaliação de desempenho do modelo, também é calculado e apresentado graficamente.

7 Redes Neurais Híbridas: CNN/LSTM

O código-fonte `cnn_lstm.py` implementa uma abordagem híbrida para modelagem de séries temporais, utilizando redes neurais convolucionais (CNN) e o método "Long Short-Term Memory" (LSTM). Primeiro, o conjunto de dados é escalonado e reformatado com a função `create_dataset` para ser compatível com o modelo LSTM. Em seguida, um modelo de rede neural sequencial contendo uma camada convolucional, uma camada LSTM e uma camada de saída é criado utilizando a função `create_model`. Este modelo é treinado ao longo de 30 épocas com dados de treino e validação. Após o treinamento, o modelo é usado para prever os valores desejados para os conjuntos de treinamento e validação. As perdas durante o treinamento e as previsões são visualizadas utilizando as funções `plot_history` e `plot_prediction`.

8 Redes Neurais Híbridas: CNN/RNN

O arquivo `cnn_rnn.py` contém uma implementação de um modelo híbrido de Convolutional Neural Network (CNN) e Recurrent Neural Network (RNN) para analisar séries temporais. Primeiro, os dados são pré-processados, e são criados objetos chamados de "lags" (desfasagens temporais) dos dados da série temporal. Posteriormente, esses dados são divididos em sequências de entrada/saída usando a função `split_sequences`. Em seguida, um modelo híbrido de CNN/LSTM é criado com a função `create_model`, que é treinado usando as sequências de entrada/saída. O modelo é avaliado e suas previsões comparadas aos dados reais, que são visualizados utilizando as funções `plot_prediction` e `plot_error`. Isso permite ver a precisão das previsões do modelo e o erro quadrático médio ao longo do treinamento.