UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE COMPUTAÇÃO

APRENDIZAGEM SUPERVISIONADO
MO432

TRABALHO Nº 03

SÉRIES TEMPORAIS APLICADAS NA ESTIMAÇÃO DO OURO

José Ítalo Da Costa Silva (RA 265682)

Gian Franco Joel Condori Luna (RA 234826)

Maria Fernanda Tejada Begazo (RA 197488)

Junho
2021

Contents

1 Introdução

Algoritmos baseados em aprendizagem supervisionada necessitam de dois componentes: as características do dados (entradas) e os rótulos correspondentes a cada padrão de entrada (saída desejada). Nesta aplicação, tem-se como entrada um conjunto de pares (X,Y) e, espera-se que o sistema aprenda a fazer mapeamento de X para Y. Ou seja, busca-se obter uma função f tal que $f(X) = Y_{\text{pred}}$, de forma que o valor estimado Y_{pred} se aproxime do valore real Y.

Uma série temporal é uma sequência de observações de uma variável ao longo do tempo [1]. De modo que, a entrada do sistema corresponde a uma sequência de dados numéricos em ordem cronológica, geralmente vistos em intervalos uniformes. Estes intervalos são determinados por meio de janelas deslizantes (*sliding window*).

Uma janela deslizante trata-se de uma pequena janela que processa os dados que vê e gera um valor resumido. Nesse contexto, a parte “deslizante” decorre do fato da janela ser, geralmente, muito menor que os dados que visualiza, esta precisa passar por todos os dados em sequência. Abaixo, na Figura 1 tem-se uma demonstração de janela deslizante retirada do arquivo "ouro2.csv".

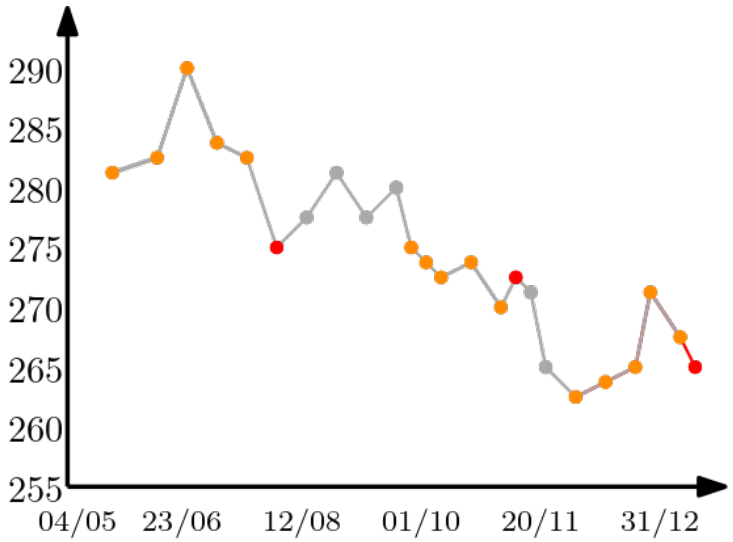

Figure 1: Exemplo de uma janela deslizante

Conforme observa-se a demonstração de janela deslizante da Figura 1, na coluna X tem-se o período (dia/mês) e em Y o valor. Em cinza tem-se o comportamento dos dados, os pontos na cor laranja fazem parte da respectiva janela deslizante, e os pontos destacados em vermelhos correspondem ao resultado.

2 Algoritmos e Codificação

O presente trabalho faz uso dos algoritmos: Regressão linear, LSTM (do inglês, Long Short Term Memory) e MLP (do inglês, Multilayer Perceptron).

2.1 Tratamentos Iniciais

Para implementação fez-se uso do arquivo "ouro2.csv" disponibilizado nesta disciplina. Este arquivo contém o preço semanal do ouro em reais, dólar e euros, de 18/06/2000 à 13/06/2021. Ademais, os tratamentos apicados foram a remoção do atributo data (formato DD/MM/YYYY) e separação das 100 entradas mais recentes.

Neste trabalho usa-se Classificação para prever se preço do ouro vai subir ou descer na próxima semana através da acurácia. Assim como Regressão, focada em prever o preço do ouro na próxima semana com uso do RMSE.

Além disso, realizou-se um pré-processamento *Min-Max* (biblioteca do sklearn) para o preço do ouro. Com isso, nossos dados vão manter-se na classe de 0 à 1. Na etapa de treinamento separou-se os dados em dois grupos: O grupo de validação, que detém as 100 entradas mais recentes e o grupo de aprendizagem.

2.2 Algoritmo: Regressão Linear

A regressão Linear trata-se de uma técnica que visa estimar o valor condicional (valor esperado) de uma variável y, dados os valores de algumas outras variáveis x. Ou seja, busca prever comportamentos com base na associação entre duas variáveis que geralmente possuem uma boa correlação.

Se implemento uma função pra obter a janela deslizante (*create_dataset*) na qual separamos nossos dados em pequenos blocos e o resultado é determinado por o valor seguente do bloco. Pra a parte da regressão linear, se uso a implementação do skelearn, o qual precisa como entrada os blocos determinados pela janela deslizante.

```
from sklearn.linear_model import LinearRegression
import numpy as np

def create_dataset(dataset, look_back=1):
	dataX, dataY = [], []
	for i in range(len(dataset)-look_back-1):
		a = dataset[i:(i+look_back), 0]
		dataX.append(a)
		dataY.append(dataset[i + look_back, 0])
	return np.array(dataX), np.array(dataY)


look_back = bestlb
trainX, trainY = create_dataset(Xtrain, look_back)
testX, testY = create_dataset(Xtest, look_back)

modelo_est = LinearRegression()
model = modelo_est.fit(trainX, trainY)
```

2.3 Algoritmo: Rede de Memória de Longo Prazo (LSTM)

O *Long short-term memory* ou rede de memória de longo prazo é um tipo especial de modelo recorrente e profundo de redes neurais, capaz de guardar informações de longo prazo. Portanto, esta implementação é adequada para classificar, processar e fazer previsões com base em dados de série temporal. Deste modo, o LSTM faz o mesmo que uma rede neural recorrente (RNN), mas com uma memória maior, fato este que lhe possibilita trabalhar com muitas tarefas.

O LSTM, assim como as redes neurais recorrentes têm a forma de uma cadeia de módulos repetitivos de rede neural. Nas RNNs padrão o módulo de repetição terá uma única camada de $tanh$. Enquanto que no LSTMs, o módulo de repetição em um LSTM contém quatro camadas de interação. Nesse sentido, os pontos de memória de uma rede LSTM são denominadas células, estas são capazes de carregar informações até o final de uma sequência ou identificar as informações que devem ser esquecidas pela rede a partir de alguma etapa de processamento.

A Figura 2 demonstra a estrutura temporal das redes LSTM. Nesta, a LSTM opera usando três portas (gates): entrada, esquecimento e saída que são denotadas, respectivamente, como I_t, f_t e O_t. De modo que, o gate de entrada decide se a informação de entrada será memorizada na célula ou não, o gate de saída define se a informação será descartada no momento t. Bem como o estado de processamento será memorizado na célula e os dados de saída serão processados pelo gate de saída. Deste modo, com o design de células demonstrado na Figura 2 e o funcionamento básico descrito acima, as redes LSTM conseguem aprender dependências de longo prazo a partir de dados com estrutura temporal [2].

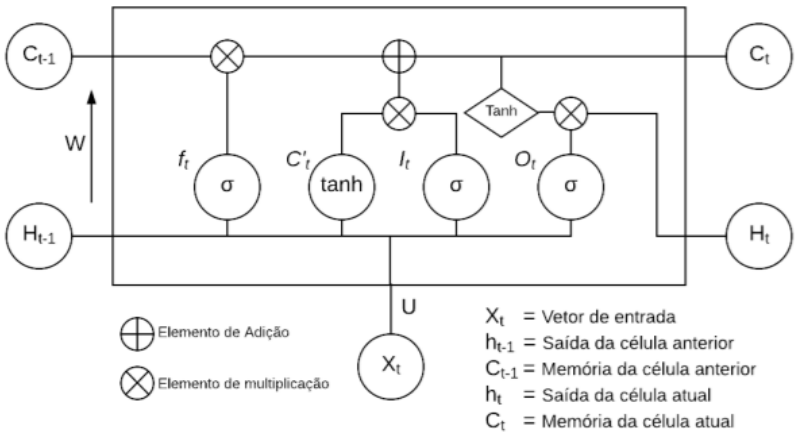

Figure 2: Estrutura LSTM

Na codificação implementada, o modelo da rede adotado é composto por uma camada LSTM que recebe os dados do ouro, já particionados pelas janelas deslizantes, de maneira que o tamanho de cada entrada esta determinado pela janela deslizante (*look_back*), depois temos uma camada densa de neurônios. Nossa rede esta usando o modelo de otimização *adam* e o erro esta determinado pelo Erro Médio Quadrático (RME), conforme verifica-se no trecho de código abaixo.

```
1 from keras.models import Sequential
2 from keras.layers import Dense
3 from keras.layers import LSTM
4
```

```
model = Sequential()
model.add(LSTM(bestNeurons, input_shape=(1, look_back)))
model.add(Dense(1))
model.compile(loss='mean_squared_error', optimizer='adam')
```

2.4 Algoritmo: Perceptron Multicamadas (MLP)

O perceptron multicamadas (MLP) é uma rede neural com mais de uma camada de neurônios em alimentação direta. Sendo esta, composta por camadas de neurônios ligadas entre si por sinapses com pesos. A propósito, o MLP têm capacidade para resolver problemas que não são linearmente separável, que é a principal limitação de rede com uma única camada. Para o aprendizado desta rede, basta realizar a retro-propagação do erro e, em seguida analisar se deve-se ou não trocar os pesos sinápticos após processar cada elemento por meio do erro entre a saída esperada e a saída da rede. A seguir, na Figura 3 tem-se a exemplificação de uma rede MLP com as camadas de entrada, camadas ocultas e camadas de saída [3].

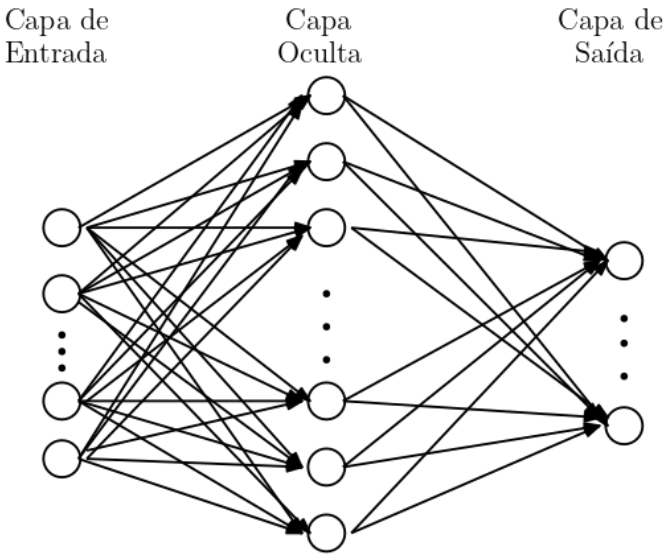

Figure 3: Estrutura perceptron Multicamada

O modelo da rede implementada é composto por uma camada densa com uma ativação *relu* que recebe os dados do ouro, já particionados pelas janelas deslizantes. De modo que, o tamanho de cada entrada está determinado pela janela deslizante (*look_back*), depois tem-se uma camada densa de neurônios. A rede implementada usa o modelo de otimização *adam* e o erro é determinado pelo Erro Médio Quadrático (RME), como verifica-se no trecho de código abaixo.

```
from keras.models import Sequential
from keras.layers import Dense

model = Sequential()
model.add(Dense(bestNeurons, activation='relu', input_shape=(1, look_back)))
model.add(Dense(1))
```

```
7  model.compile(optimizer='adam', loss='mse')
```

3 Resultados

3.1 Etapa de treinamento

Esta etapa é baseada nos dados de treinamento para busca dos hiper-parâmetros em cada técnica realizada.

- **LSTM**

  Ao tratar do LSTM, foram testados os seguintes hiperparâmetros:

  - *lb = [4, 5, 10, 15, 20, 25, 30]*
  - *neuronios = [5, 7, 10, 12]*
  - *epocas = [50, 75, 100]*

  Com isso, observou-se que para obter os dados da predição deve-se usar o *r2_score*, pois permite trabalhar com saídas de valores contínuos. Portanto, nossos melhores hiperparâmetros foram:

  - *lb = 20*
  - *neuronios = 5*
  - *epocas = 75*

Ao executar o algoritmo, os resultados obtidos foram: um RMSE de 61 e uma acurácia de 78%. Ademais, na Figura 4, são mostrados os dados de predição (*prediction*), em vermelho, comparados com os dados originais(*True*), em azul.

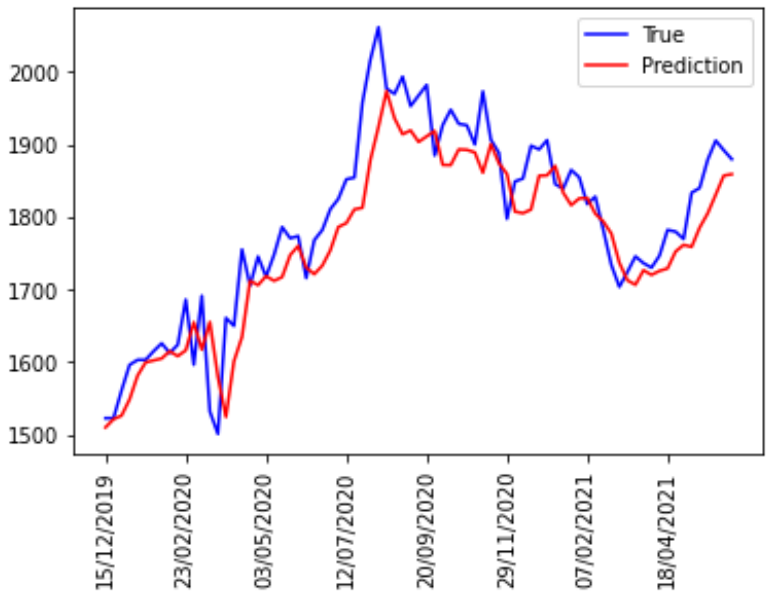

Figure 4: Resultados da LSTM dos últimos 100 dados

- **MLP**

Na MLP, ao buscar os melhores hiperparâmetros, trabalhou-se com os seguintes valores:

  - *lb = [4, 5, 10, 15, 20, 25, 30]*
  - *neuronios = [5, 7, 10, 12]*
  - *epocas = [50, 75, 100]*

Então, os melhores hiperparâmetros obtidos foram:

  - *lb = 25*
  - *neuronios = 12*
  - *epocas = 100*

Como resultado obteve-se: um RMSE de 64 e uma acurácia de 71%. Na Figura 5 são mostrados os dados de predição (*prediction*), em vermelho, comparados com os dados originais(*True*), em azul.

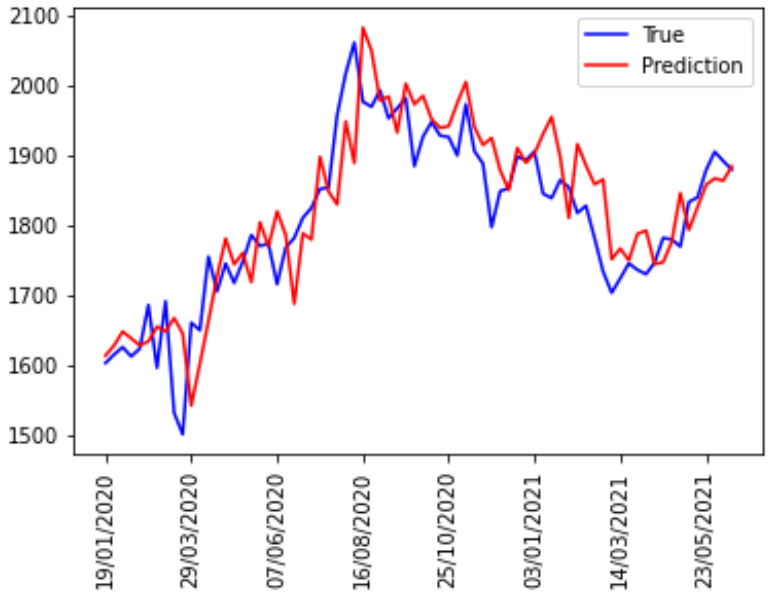

Figure 5: Resultados do MLP dos últimos 100 dados

- **Regressão Linear**

Ao aplicar a regressão linear procuramos o melhor tamanho pra a janela deslizante, o classificador de pesquisa é o mesmo que das outras técnicas anteriores. Dest a forma, o resultado alcançado foi um RMSE de 45 e uma acurácia de 91% e com uma janela de 5 (vide Figura 6).

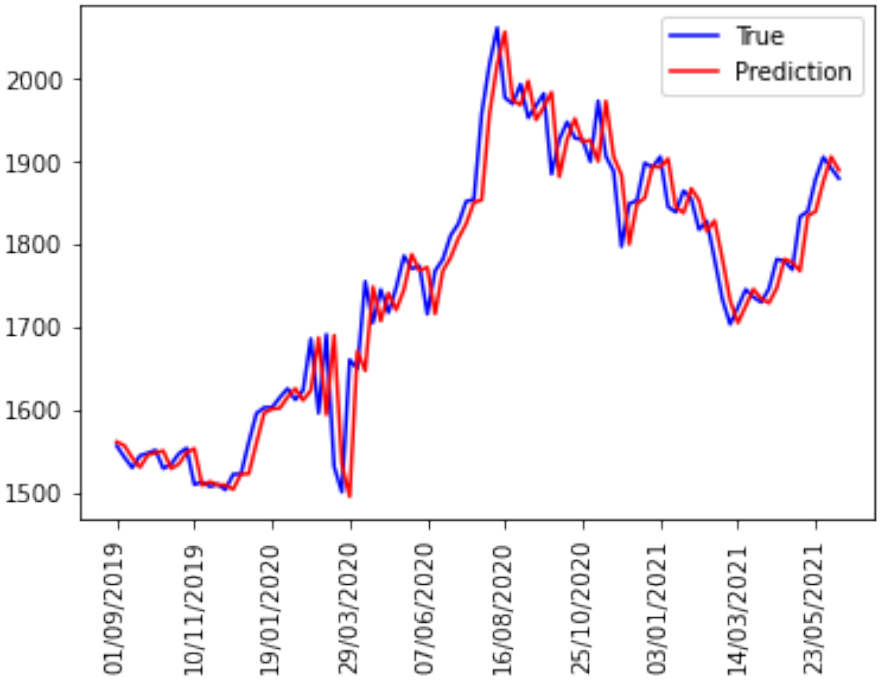

Figure 6: Resultados da regressão linear dos últimos 100 dados

3.2 Resultados Finais

Na tabela 1 observa-se que o melhor algoritmo para os dados do ouro é uma regressão linear, já que sua acurácia é 91% e RMSE de 45 aproximadamente. Seguida pela técnica do LSTM e finalmente o MLP, todos aplicados conforme as características determinadas anteriormente. Ademais, na parte do treinamento, a acurácia e o RMSE apresentavam um valor muito bom, mas quando foi testado com dados não vistos no treinamento, diminuiu-se os valores destas métricas.

	Treino		**Teste**	
	RMSE	**ACC (%)**	**RMSE**	**ACC (%)**
LSTM	26.87	100	61.48	78
MLP	38,02	99	64.03	71
Regressão Linear	25.08	100	44.94	91

Images 1: Resultados de treino e teste

4 Conclusão

Neste trabalho para obtenção de um classificador do parâmetro para o tamanho da janela deslizante fizemos testes com valores de 50 e 2, pois buscou-se observar se era melhor focar-se no passado ou não. Neste caso, para o valor 50 o erro obtido mostrou-se muito alto, por isso escolhemos valores pequenos para a procura do melhor tamanho para a janela.

Por conseguinte, por meio desta aplicação pode-se concluir que nas séries temporais têm-se uma acurácia muito baixa em comparação com outras aplicações vistas nesta disciplina. Pois, em alguns

períodos do tempo os valores parecem ter uma tendencia aleatória, dificultando assim a predição no tempo. Portanto, conclui-se que, ao observar estas três técnicas, obteve-se as melhores aplicações ao usar janelas de tamanho pequeno.

References

[1] J. M. Wooldridge, *Introductory econometrics: A modern approach*. Cengage learning, 2015.

[2] G. S. de Melo, “Aplicação de aprendizado de máquina para previsão de fluxo de caixa em atms.”

[3] W. Oliveira, D. Gaia, F. Monteiro, B. Rodrigues, J. Vieira, and U. Bezerra, “Comparação dos algoritmos c4. 5 e mlp aplicados a avaliação da segurança dinâmica e ao controle preventivo no contexto da estabilidade transitória,” *The 10th CLAGTEE, Vina del Mar*, 2013.