Bitcoin Price Prediction Using Recurrent Neural Networks and LSTM

ADVANCED MACHINE LEARNING PROGRAMMING PROJECT PYTHON STOCK TRADING STRUCTURED DATA SUPERVISED TIME SERIES FORECASTING

This article was published as a part of the Data Science Blogathon

Index

1. Introduction

Recurrent neural networks (RNN) are the state-of-the-art algorithm for sequential data and are used by Apple's Siri and Google's voice search. It is an algorithm that remembers its input due to its internal memory, which makes the algorithm perfectly suited for solving machine learning problems involving sequential data. It is one of the algorithms that have great results in deep learning. In this article, it is discussed how to predict the price of Bitcoin by analyzing the information of the last 6 years. We implemented a simple model that helps us better understand how time series works using Python and RNNs.

2. Understanding Bitcoin

Bitcoin is a cryptocurrency that was created in January 2009. It is the world's most valuable cryptocurrency and is traded on over 40 exchanges around the world, accepting over 30 different currencies. As a currency, Bitcoin offers a new opportunity for price forecasting as it has high volatility, which is much higher compared to traditional currencies.

The bitcoin system is a set of decentralized nodes that run the bitcoin code and store its blockchain. Metaphorically, a blockchain can be considered a collection of blocks. In each block, there is a collection of transactions. Because all the computers running the blockchain has the same list of blocks and transactions, and can transparently see these new blocks being filled with new bitcoin transactions, no one can cheat the system.

Bitcoin uses peer-to-peer technology to facilitate instant payments. Miners are responsible for processing transactions on the blockchain and are driven by repo fees.

The way bitcoin works are key to understanding why it is so popular. Unlike other investments, cryptocurrency is not tied to physical assets or the US dollar. Its primary purpose is to allow two people anywhere to exchange value directly. What this means is that there is no centralized controlling this network. There is no government, no central bank that can shut down or arbitrarily raise or lower the value.

It will be interesting to see to which degree central banks start to digitize their own currencies. As financial systems become more digital, it's leading to bitcoin more mainstream, but the digital currency's resurgence is also closely tied to the state of global finance.

Every time someone makes a transaction, a unique encrypted signature is added to the ledger for verification

3. Technologies used

1. Recurrent Neural Networks

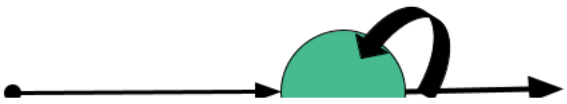
RNNs are a robust and powerful type of neural network and are considered one of the most professional algorithms because they are the only ones with internal memory.

Recurrent neural networks were first created in the 1980s, but only in recent years has their true potential been realized. The increase in its computational power, along with the gigantic amounts of data we now have to work with, and the invention of short-term memory (LSTM) in the 1990s, has really brought RNNs to the fore.

The algorithm performs very well for sequential data such as time series, speech, text, financial data, audio, video, weather, and more. RNNs are able to form a much deeper understanding of a sequence and its context compared to other algorithms.

In an RNN, the information goes through a cycle. When making a decision, it considers the current input and also what it has learned from the inputs it has received previously.

The image below illustrates how the flow of information works in the RNN algorithm.



2. Long Short-Term Memory (LSTM)

Long short-term memory networks are an extension of recurrent neural networks, which basically extend the memory. Therefore it is well suited to learn from important experiences that have very long time lags in between.

LSTMs enable RNNs to remember inputs over a long period of time. This is because LSTMs contain information in a memory, much like the memory of a computer. The LSTM can read, write and delete information from its memory.

In an LSTM you have three gates: input, forget and output gate. These gates determine whether or not to let new input in (input gate), delete the information because it isn't important (forget gate), or let it impact the output at the current timestep (output gate). Below is an illustration of an RNN with its three gates:

The gates in an LSTM are analog in the form of sigmoids, meaning they range from zero to one. The fact that they are analog enables them to do backpropagation.

4. Getting real-time cryptocurrency data (bitcoin)

The data is collected the current data for Bitcoin from [Yahoo Finance]

```
import numpy as np import matplotlib.pyplot as plt import pandas as pd from sklearn.preprocessing import MinMaxScaler


data = pd.read_csv('BTC-USD.csv', date_parser = True) data.tail()
```

```
data_training = data[data['Date']< '2020-01-01'].copy() data_training
```

```
data_test = data[data['Date']< '2020-01-01'].copy() data_test
```

```
training_data = data_training.drop(['Date', 'Adj Close'], axis = 1) training_data.head()
```

5. Normalization

The first step we will take to our data is to normalize its values. The goal of normalization is to change the values of numeric columns in the data set to a common scale, without distorting differences in the ranges of values.

```
MinMaxScaler is used to normalize the data scaler = MinMaxScaler() training_data = scaler.fit_transform(training_data) training_data
```

```
X_train = [] Y_train = []
```

```
training_data.shape[0]
```

```
for i in range(60, training_data.shape[0]):
```

```
X_train.append(training_data[i-60:i])
```

```
Y_train.append(training_data[i,0])
```

```
X_train, Y_train = np.array(X_train), np.array(Y_train) X_train.shape
```

6. Predict the price of cryptocurrency using LSTM neural network (deep learning)

This is the model-building stage. Finding the right model is an art, and it will take several tweaks and attempts to find the right layers and hyperparameters for each one.

The model building is quite simple and standard for this type of problem.

Training this model is something you can do even without a GPU, the amount of data is very low and the network architecture is very simple. When it comes to more advanced models with more granular information, it can take hours or days to train.

```
from tensorflow.keras import Sequential


from tensorflow.keras.layers import Dense, LSTM, Dropout


#Initialize the RNN


model = Sequential() model.add(LSTM(units = 50, activation = 'relu', return_sequences = True, input_shape = (X_train.shape[1], 5)))


model.add(Dropout(0.2)) model.add(LSTM(units = 60, activation = 'relu', return_sequences = True))


model.add(Dropout(0.3)) model.add(LSTM(units = 80, activation = 'relu', return_sequences = True))


model.add(Dropout(0.4)) model.add(LSTM(units = 120, activation = 'relu'))


model.add(Dropout(0.5)) model.add(Dense(units =1))


model.summary()
```

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


history= model.fit(X_train, Y_train, epochs = 20, batch_size =50, validation_split=0.1
```

```
loss = history.history['loss'] val_loss = history.history['val_loss'] epochs = range(len(loss)) plt.figure() plt.plot(epochs, loss, 'b', label='Training loss') plt.plot(epochs, val_loss, 'r', label='Validation loss') plt.title("Training and Validation Loss") plt.legend() plt.show()
```

7. Test Dataset

```
part_60_days = data_training.tail(60) df= part_60_days.append(data_test, ignore_index = True) df = df.drop(['Date', 'Adj Close'], axis = 1) df.head()
```

```
inputs = scaler.transform(df) inputs
```

```
X_test = []
```

```
Y_test = []
```

```
for i in range (60, inputs.shape[0]):
```

```
X_test.append(inputs[i-60:i]) Y_test.append(inputs[i, 0])
```

```
X_test, Y_test = np.array(X_test), np.array(Y_test) X_test.shape, Y_test.shape
```

```
Y_pred = regressor.predict(X_test) Y_pred, Y_test
```

```
scaler.scale_
```

```
scale = 1/5.18164146e-05
```

```
Y_test = Y_test*scale Y_pred = Y_pred*scale
```

Y_pred

Y_test

```
plt.figure(figsize=(14,5)) plt.plot(Y_test, color = 'red', label = 'Real Bitcoin Price') plt.plot(Y_pred, color = 'green', label = 'Predicted Bitcoin Price') plt.title('Bitcoin Price Prediction using RNN-LSTM') plt.xlabel('Time') plt.ylabel('Price') plt.legend() plt.show()
```

Conclusion

RNNs and LSTM are excellent technologies and have great architectures that can be used to analyze and predict time-series information. The focus of the article was to implement a simple model, if you are interested in the subject, try different things and want to play with hyperparameters and layers.

The media shown in this article on Bitcoin Price Prediction are not owned by Analytics Vidhya and is used at the Author's discretion.

Article Url - https://www.analyticsvidhya.com/blog/2021/05/bitcoin-price-prediction-using-recurrent-neural-networks-and-lstm/

ana_lucia