Chris Liverani 发表在 Unsplash 杂志上的照片

2017 年，由于加密货币市值连续几个月呈指数增长，其受欢迎程度飙升。加密货币的价格在 2018 年 1 月达到 8000 多亿美元的峰值。

尽管机器学习已经成功地通过一系列不同的时间序列模型来预测股市价格，但它在预测加密货币价格方面的应用却非常有限。其背后的原因是显而易见的，因为加密货币的价格取决于许多因素，如技术进步、内部竞争、市场交付压力、经济问题、安全问题、政治因素等。如果采取明智的投资策略，它们价格的高波动性将带来巨大的利润。不幸的是，由于缺乏指数，与股市等传统金融预测相比，加密货币的预测相对较难。

在这篇文章中，作者将用总共四个步骤来预测加密货币的价格：

- 获取实时加密货币数据
- 准备训练和测试数据
- 用 LSTM 神经网络预测货币价格
- 可视化预测结果

挑战

使用数据集中的所有交易特征（如价格、交易量、未平仓、高值和低值）预测加密货币价格。

数据

数据集可以从 CryptoCompare 网站下载。

数据集总共包含 5 个特征。具体情况如下：

1. Close Price：指当日货币的市场收盘价
2. High Price：当天货币的最高价格
3. Low Price：是当天货币的最低价

4. Open Price：当日货币的市场公开价格

5. Volume：当天交易的货币量

代码在哪里？

不费吹灰之力，让我们从代码开始。完整的 github 项目可以在这里找到：https://github.com/abhinavsagar/Cryptocurrency-Price-Prediction

从加载所需的所有库和依赖项开始：

```
import json

import requests

from keras.models import Sequential

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

import matplotlib.pyplot as plt

import numpy as np

import pandas as pd

import seaborn as sns

from sklearn.metrics import mean_absolute_error

%matplotlib inline
```

我使用了加拿大的汇率，并将实时数据存储到 pandas
数据框中，将字符串日期时间转换为 Python 日期时间对象。这是必需的，因为文件中的日期时间对象是作为字符串对象读取的，对字符串而不是日期时间对象执行时间差之类的操作非常容易。

```
endpoint = 'https://min-api.cryptocompare.com/data/histoday'
```

```
res = requests.get(endpoint + '?fsym=BTC&tsym=CAD&limit=500')

hist = pd.DataFrame(json.loads(res.content)['Data'])

hist = hist.set_index('time')

hist.index = pd.to_datetime(hist.index, unit='s')

target_col = 'close'
```

让我们看看数据集的所有交易特性，如价格、成交量、开盘、高价、低价，是什么样子的。

```
hist.head(5)
```

接下来，我将数据分为两组：训练集和测试集，它们分别占据 80% 和 20% 的数据量。这里只是为了这个项目才这样做的，在实际项目中，你应该始终将数据分为训练、验证、测试三个数据集（占比可以分别为 60%、20%、20%）。

```
def train_test_split(df, test_size=0.2):

split_row = len(df) - int(test_size * len(df))

train_data = df.iloc[:split_row]

test_data = df.iloc[split_row:]

return train_data, test_datatrain, test = train_test_split(hist,
test_size=0.2)
```

现在，让我们使用以下代码绘制加密货币（加元）价格随时间变化的曲线：

```
def line_plot(line1, line2, label1=None, label2=None, title='', lw=2):

fig, ax = plt.subplots(1, figsize=(13, 7))
```

```
ax.plot(line1, label=label1, linewidth=lw)

ax.plot(line2, label=label2, linewidth=lw)

ax.set_ylabel('price [CAD]', fontsize=14)

ax.set_title(title, fontsize=16)

ax.legend(loc='best', fontsize=16)

line_plot(train[target_col], test[target_col], 'training', 'test', title='')
```

我们可以观察到，2018 年 12 月至 2019 年 4 月期间，加密货币的价格明显下降；2019 年 4 月至 2019 年 8 月，其价格持续上涨，在 7、8 月份出现波动；从 2019 年 9 月起，价格不断下降。在这次价格波动中，值得注意的是，加密货币在冬季价格较低，在夏季价格上涨。但是，由于数据集太小，这不能得出可以普遍适用的结论。同样，对于加密货币，很难概括任何有价值的结论。

接下来，我写了几个函数来规一化这些值。规一化是机器学习中常用的一种数据准备技术。规一化的目标是将数据集中数值列的值更改为公共比例，而不会扭曲值范围中的差异。

```
def normalise_zero_base(df):

return df / df.iloc[0] - 1

def normalise_min_max(df):

return (df - df.min) / (data.max - df.min)
```

接下来，我写了一个函数来提取大小为 5 的窗口的数据，如下代码所示：

```
def extract_window_data(df, window_len=5, zero_base=True):

window_data =
```

```
for idx in range(len(df) - window_len):

tmp = df[idx: (idx + window_len)].copy

if zero_base:

tmp = normalise_zero_base(tmp)

window_data.append(tmp.values)

return np.array(window_data)
```

我继续编写函数，以准备数据的格式，稍后将其输入神经网络。我使用了和前面相同的概念，将数据分成两组：训练集和测试集，它们分别占总数据的 80% 和 20%，代码如下：

```
def prepare_data(df, target_col, window_len=10, zero_base=True, test_size=0.2):

train_data, test_data = train_test_split(df, test_size=test_size)

X_train = extract_window_data(train_data, window_len, zero_base)

X_test = extract_window_data(test_data, window_len, zero_base)

y_train = train_data[target_col][window_len:].values

y_test = test_data[target_col][window_len:].values

if zero_base:

y_train = y_train / train_data[target_col][:-window_len].values - 1

y_test = y_test / test_data[target_col][:-window_len].values - 1

return train_data, test_data, X_train, X_test, y_train, y_test
```

LSTM

LSTM 的工作原理是使用特殊的门允许每个 LSTM 层从前一层和当前层获取信息。数据通过多个门（如遗忘门、输入门等）和各种激活函数（如 tanh 函数、relu 函数）并通过 LSTM 单元。它的主要优点是允许每个 LSTM
单元在一定时间内记住这个模式。需要注意的是，LSTM
能够记住重要的信息，同时忘记不相关的信息。LSTM 体系结构如下所示：

LSTM 体系架构

现在让我们建立模型。序列模型用于将所有层（输入、隐藏和输出）堆叠起来。该神经网络由一个 LSTM 层、20%
脱落层和一个具有线性激活函数的稠密层组成。我使用 Adam
作为优化器，使用均方误差作为损失函数来编译模型。

```
def build_lstm_model(input_data, output_size, neurons=100,
activ_func='linear', dropout=0.2, loss='mse', optimizer='adam'):

model = Sequential

model.add(LSTM(neurons, input_shape=(input_data.shape[1],
input_data.shape[2])))

model.add(Dropout(dropout))

model.add(Dense(units=output_size))

model.add(Activation(activ_func))

model.compile(loss=loss, optimizer=optimizer)

return model
```

接下来，我设置一些参数供以后使用。这些参数是：随机数种子，窗长度，测试集大小，第一层神经元数量，批量大小，损失和优化器等。

```
np.random.seed(42)
```

```
window_len = 5

test_size = 0.2

zero_base = True

lstm_neurons = 100

epochs = 20

batch_size = 32

loss = 'mse'

dropout = 0.2

optimizer = 'adam'
```

现在让我们使用输入 x_train 和标签 y_train 来训练模型。

```
train, test, X_train, X_test, y_train, y_test = prepare_data(

hist, target_col, window_len=window_len, zero_base=zero_base,
test_size=test_size)model = build_lstm_model(

X_train, output_size=1, neurons=lstm_neurons, dropout=dropout,
loss=loss,

optimizer=optimizer)

history = model.fit(

X_train, y_train, epochs=epochs, batch_size=batch_size, verbose=1,
shuffle=True)
```

让我们来看看 20 个 epoch 的模型训练快照。

神经网络的训练

我用平均绝对误差（MAE）作为评价指标。选择 MAE
而不是均方根误差（RMSE）的原因是 MAE 更易于解释。RMSE 并不单独描述平均误差，因此更难理解。因为我们希望即使是对完全不懂技术的读来说，模型也可以很容易理解，因此 MAE 看起来是一个更好的选择。

平均绝对误差

平均绝对误差测量一组预测中误差的平均大小，而不考虑它们的方向。它是实际观测值和预测观测值之间的绝对差在测试样本上的平均值，其中所有的个体差异具有相同的权重。

```
targets = test[target_col][window_len:]

preds = model.predict(X_test).squeeze

mean_absolute_error(preds, y_test)

# 0.027955859325876943
```

获得 MAE 值看起来不错。最后，让我们使用以下代码绘制实际价格和预测价格：

```
preds = test[target_col].values[:-window_len] * (preds + 1)

preds = pd.Series(index=targets.index, data=preds)

line_plot(targets, preds, 'actual', 'prediction', lw=3)
```

结论

在本文中，我演示了如何使用 LSTM 神经网络实时预测加密货币价格。我使用了四个步骤：获取实时货币数据、准备数据进行训练和测试、使用 LSTM 神经网络预测价格和可视化预测结果。可以随意使用超参数或尝试不同的神经网络结构以获得更好的结果。

via：https://towardsdatascience.com/cryptocurrency-price-prediction-using-

deep-learning-70cfca50dd3a

雷锋网雷锋网雷锋网