Report
Prediction of Cryptocurrency with RNN and SVM

Tianshu Bao, Yingqi Li, Pengfei Wang

Introduction
We used two models (RNN and SVM) to predict three kinds of cryptocurrencies (Bitcoin, Ethereum and Litecoin) in the first week of December 2017 and compared the reliability of the two models.

Data
How was the data collected?
We got up-to-date cryptocurrency prices from the website:
https://finance.yahoo.com/quote/BTC-USD/history?p=BTC-USD
https://finance.yahoo.com/quote/ETH-USD/history?p=ETH-USD
https://finance.yahoo.com/quote/LTC-USD/history?p=LTC-USD

How is the data composed?
We compose the price of Bitcoin (BTC), Ethereum (ETH) and Litecoin (LTC) in csv format.

What does the data describe?
The csv file describes the volumes, dates, open, high and low prices.

Background

Model 1 RNN
A recurrent neural network (RNN) is a class of artificial neural network where connections between units form a directed graph along a sequence. This allows it to exhibit dynamic temporal behavior for a time sequence. RNNs can use their internal state (memory) to process sequences of inputs.

Keras model
Keras is an open source neural network library written in Python. Keras contains numerous implementations of commonly used neural network building blocks such as layers, objectives, activation functions, optimizers, and a host of tools to make working with image and text data easier.

Keras Sequential model

The Sequential model is a linear stack of layers. We can create a Sequential model by passing a list of layer instances to the constructor. In our approach, we initialize it by regressor=Sequential().

We can also simply add layers via the *.add()* method. In our approach, we add input layer and LSTM layer.

Specifying the input shape

When specifying the input shape, we Pass an input_shape argument to the first layer. This is a shape tuple (a tuple of integers or None entries, where None indicates that any positive integer may be expected). In input_shape, the batch dimension is not included.

Some 2D layers, such as Dense, support the specification of their input shape via the argument input_dim.

If we need to specify a fixed batch size for your inputs (this is useful for stateful recurrent networks), you can pass a batch_size argument to a layer. If you pass both batch_size=32 and input_shape=(6, 8) to a layer, it will then expect every batch of inputs to have the batch shape (32, 6, 8).

Compilation

Before training a model, you need to configure the learning process, which is done via the compile method. It receives three arguments:

An optimizer. This could be the string identifier of an existing optimizer or an instance of the Optimizer class.

A loss function. This is the objective that the model will try to minimize. It can be the string identifier of an existing loss function or it can be an objective function.

A list of metrics. For any classification problem you will want to set this to metrics=['accuracy']. A metric could be the string identifier of an existing metric or a custom metric function.

We use *Adam optimizer* and *mean_squared_error loss function*:

Adam, an algorithm for first-order gradient-based optimization of stochastic objective functions, based on adaptive estimates of lower-order moments.

Training

Keras models are trained on Numpy arrays of input data and labels. For training a model, you will typically use the fit function.

Here, we train the model, iterating on the data in batches of 32 samples:

regressor.fit(xtrain,ytrain,batch_size=32,epochs=1000).

Model 2 SVM

In machine learning, support vector machines are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM

training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier.

SVR
A version of SVM for regression is called support vector regression (SVR). The model produced by support vector classification (as described above) depends only on a subset of the training data.

Kernel
SVM can be divided into two categories: (a) Linearly separable training, (b) Non-linearly separable training.

Non-linearly separable training is to gain linearly separation by mapping the data to a higher dimensional space. An SVM kernel is used to transform training and testing data into a higher dimensional space, which provides better linear separation.

A kernel function is defined as follows, **K(xi,xj) = Φ(xi) Φ(xj),** where Φ(x) is a mapping function used to convert input vectors x to a desired space. Linear, polynomial, radial basis function (gaussians) and sigmoid (neural net activation function) are used as non-linear mapping functions.

In RBF kernel, the *gamma* parameter defines how far the influence of a single training example reaches, with low values meaning 'far' and high values meaning 'close'. The gamma parameters can be seen as the inverse of the radius of influence of samples selected by the model as support vectors.

The *C* parameter trades off misclassification of training examples against simplicity of the decision surface. A low C makes the decision surface smooth, while a high C aims at classifying all training examples correctly by giving the model freedom to select more samples as support vectors.

Approach

Model 1 RNN
Recurrent Neural Networks are excellent to use along with time series analysis to predict Virtual currency prediction. Time series forecasting is the use of a model to predict future values based on previously observed values.

Preprocesssing the data
The prices of all cryptocurrencies are stored in the csv file. We choose Bitcoin for instance. Firstly, we read data from csv file and convert it to 2d array. Secondly, we scale the data using normalization, from sklearn.preprocessing import MinMaxScaler. Thirdly, we reshape for Keras. Similarly, we preprocessed the ETH csv file.

Building the RNN

We import keras and its packages. We initialize the RNN and add input layer, LSTM layer and output layer. The units in LSTM is 4 and the activation is 'sigmoid'. We compile the RNN, where one parameter optimizer is 'adam', one parameter loss is 'mean_squared_error'. We fit the RNN to the training set, where batch_size is 32 and epochs is 1000. The parameters of ETH and LTC is the same with the parameters of BTC.

Making predictions

We convert the BTC, ETH and LTC data to 2D array, and get the predicted Bitcoin, Ethereum and Litecoin price of the first week of Dec 2017. And we plot and get the results.

Model 2 SVM

Preprocesssing the data

We will predict the price of day t using the prices of the day t-1, t-2 and t-3. So we can concatenate a price matrix as an input and plot the relationship between the variable x combined the prices of t-1, t-2 and t-3.

Building the SVM

We use three kinds of kernels including linear and non-linear.

After changing parameters many times, we found the most suitable parameters of these three kernels:

The parameter of linear kernel is C=1e2.

The parameter of polynomial kernel is C=1e2, degree=1.

The parameter of RBF kernel is C=1e2, gamma=0.001.

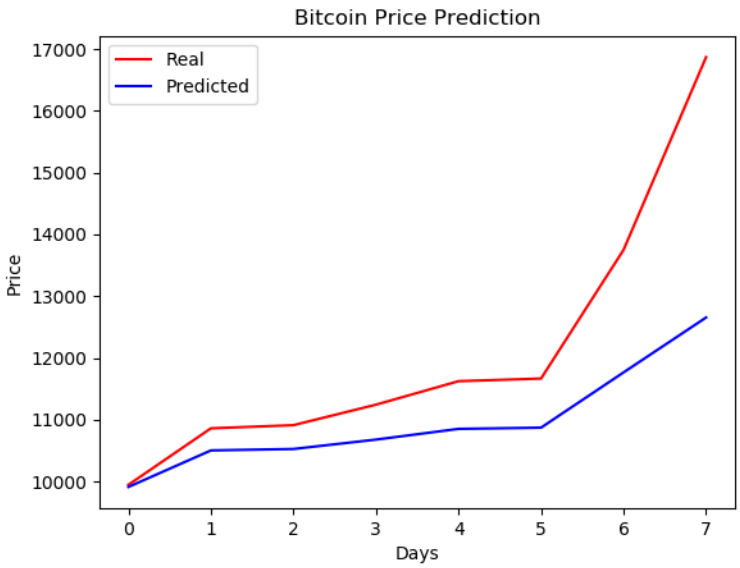
Making predictions

We reshape the data to the same format of the result run in RNN and get the predicted Bitcoin and ETH price of the first week of Dec 2017. We use three kernels (linear/polynomial/RBF) to train the data separately. And we plot and compare the results of these three kernels. Then we choose the best result using SVM to compare with the result using RNN.

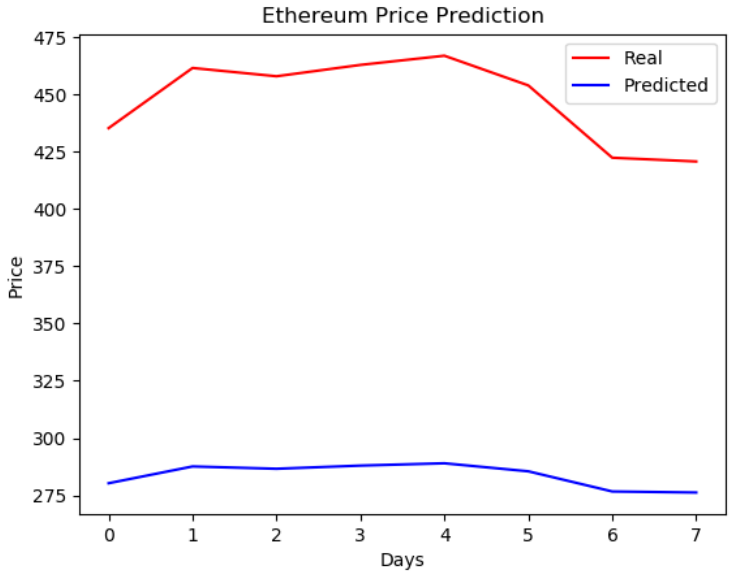
Results

Model 1 RNN

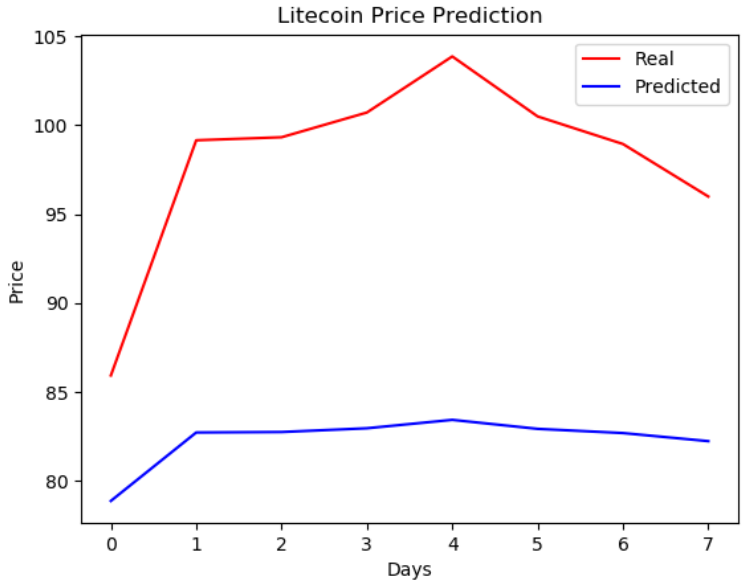
Use RNN to predict Bitcoin price



Use RNN to predict Ethereum price

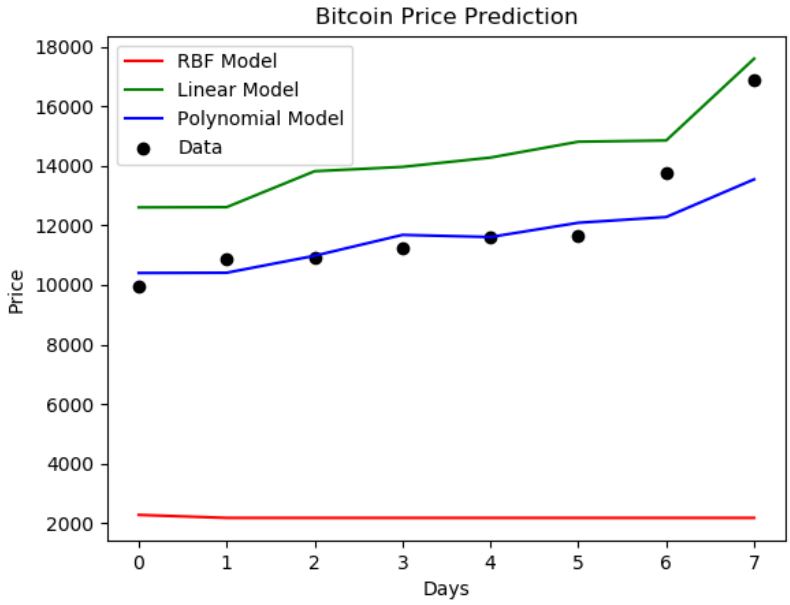


Use RNN to predict Litecoin price

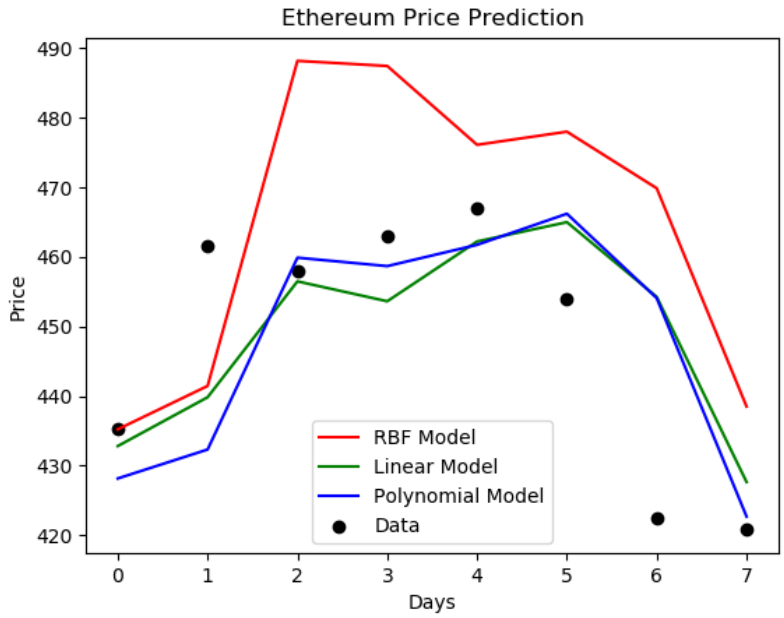


Model 2 SVM

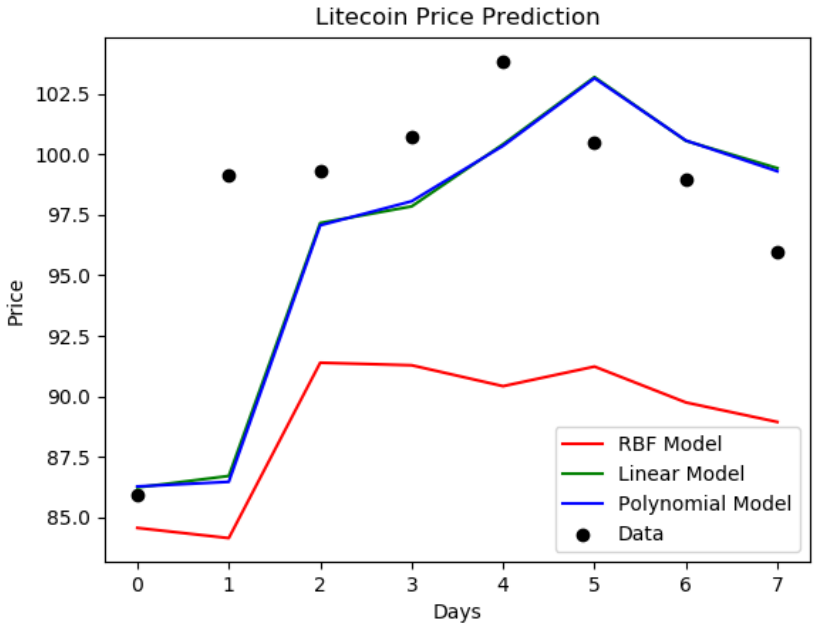
Use SVM to predict Bitcoin price



Use SVM to predict Ethereum price



Use SVM to predict Litecoin price



Discussions

1. Is the Cryptocurrency prediction using RNN (the parameters we used) reliable? Why?
Not that reliable.
Although the prediction tendency is consistent with the actual tendency, the value difference between the prediction and actual is really huge, especially Ethereum and Litecoin.
We have changed parameters continuously until we found the most suitable one:
We use optimizer 'adam', loss 'mean_squared_error', batch_size 32, epochs 1000.
Compared with the results using SVM, the Cryptocurrency prediction using RNN is less reliable.

2. Is the Cryptocurrency prediction using SVM (the parameters we used) reliable? Why?
It depends on the kernel.
In general, predictions with linear kernel and polynomial kernel are more accurate than RBF kernel. In terms of the prediction result of the first week in December, if we use linear kernel and polynomial kernel, the tendency of the predicted Cryptocurrency value is almost consistent with real Cryptocurrency value although sometimes does not fit very well. And the value differences between the predicted Cryptocurrency value and the real Cryptocurrency value with linear kernel and polynomial kernel are smaller than the difference with RBF kernel. In terms of the Bitcoin prediction, prediction with polynomial kernel fits the real value very well, which is the best prediction. In terms of the ETH prediction and Litecoin prediction, the performances of prediction with linear kernel and polynomial kernel are almost the same.
Also, we continuously change parameters until we find the most suitable parameters:
The parameter of linear kernel is C=1e2.
The parameter of polynomial kernel is C=1e2, degree=1.
The parameter of RBF kernel is C=1e2, gamma=0.001.
After analyzing predictions with these three kernels, the polynomial kernel fits the real value best of these three. Comparing prediction using SVM with polynomial kernel and prediction using RNN, the Cryptocurrency prediction using SVM with polynomial kernel is more reliable.

3. When will we invest to get the largest profit using the more reliable method?
According to our prediction using SVM with polynomial kernel, we will buy Bitcoin on December 1^{st} and sell Bitcoin on December 7^{th}. While we will buy ETH on December 1^{st} and sell ETH on December 5^{th}. And we will buy Litecoin on December 1^{st} and sell Litecoin on December 5^{th}. Although prediction using SVM with polynomial kernel does not predict the highest peak precisely, we can still get large profit using the predition.

4. Considering the lock stock situation (hold coins for a specific period, here we can define the specific period as the first week of December), which cryptocurrency should we invest to get the largest profit using the more reliable method?
Increase percentage = (Price of Dec 7^{th} – Price of Dec 1^{st})/Price of Dec 1^{st} $*$ 100%
According to the predictions using SVM with polynomial kernel, the increase of Bitcoin is about 25.7%, the increase of ETH is about -2.0%, and the increase of Litecoin is about 14.78%.
We should invest on the Bitcoin.

Conclusions

After analyzing two models (RNN and SVM) to predict three kinds of cryptocurrencies (Bitcoin, Ethereum and Litecoin) in the first week of December 2017, predictions using SVM with polynomial kernel with the parameters we used is more reliable than predictions using RNN with the parameters we used. We can use SVM (polynomial kernel) prediction results to get the largest profit when investing cryptocurrencies.