CS230

Predicting Cryptocurrency Price Fluctuation Based on Twitter, Media, and Currency Data With an LSTM-RNN

Alex K. Fine
Stanford University
afine@stanford.edu

Abstract

This algorithm is designed to provide investment insights that mitigate investor risk. I build upon prior work to predict binary price change across 100 currencies through an analysis of previous behavior, tweet data, and news data. Overall, prediction quality was varied and hard to evaluate. The model did perform better than expected, reaching 90% accuracy very short temporal regions. However, this high accuracy could be heavily attributed to training data that skewed heavily in a single direction.

1 Introduction

1.1 Motivation

Most Americans have loans, most Americans have money in the bank, and most Americans aren't investing. This is a problem.

When I arrived at college I took a part-time job to help cover some of the many expenses. After a few weeks I had a little money and knew that it should be invested, as opposed to sitting idle in a bank account. However, when I tried investing, I realized that I didn't know what I was doing. Even after many hours of research I was left lost, with strategies barely better than random guessing. This experience led me to realize that there is a serious disconnect in this country. 44.2 million Americans have graduated with a collective 1.4 trillion dollars in student loans. However, less than 10% of current college students invest, and even fewer invest strategically.

My motivation for this project stemmed from this realization. I wanted to give college students and low-income individuals access to the same investment opportunities previously only available to the hyper rich and highly educated. Additionally, I wanted to do so in a way that mitigates investor risk, and encourages students to invest.

To help students, I first knew that I would need some way of identifying promising investments. A NN could solve this problem. Next, I wanted the investment opportunities to be "cool" and appeal to college-aged students. With the most recent crypto-currency hype, this choice was clear. Lastly, the students must feel safe investing. Therefore, I decided to design my model around mitigating risk, as opposed to aggressively identify high-risk high-reward opportunities.

1.2 Model

My algorithm is an LSTM-RNN that predicts price change in the 100 most traded currencies. Its inputs are all stored per minute, and include the following:

1. Historic currency behavior
2. Aggregate Twitter sentiment for each currency
3. Aggregate news sentiment for each currency

I first execute a number of data processing and cleansing techniques as discussed in section 3. Next, I run the data through a deep-RNN with multiple LSTM cells. Lastly, a softmax layer is applied that returns a binary prediction for whether the currency's price is going to rise or fall.

2 Related work

Researchers have invested considerable time into identifying arbitrage opportunities through social media and news data. This work has yielded numerous learnings that can be grouped into two general categories:

2.1 How to best analyze tweet data

Colianni et al.[SOURCE] and Pagolu et al.[SOURCE] have both published work demonstrating tweets' high correlation with market data. In "Algorithmic Trading of Cryptocurrency Based on Twitter Sentiment Analysis"[SOURCE] by Colianni et al.[SOURCE], prediction accuracy increased from 59% to 76% after tweets were processed with the Bernoulli Naive Bayes classifier and cleansed of duplicates. Pagolu et al. demonstrated that through removing erroneous information (words) from tweets and then running sentiment classification over the tweets, they could predict market change with 70% accuracy. One of the most cited works in this sub-field is by Stenqvist et al.[SOURCE]. These researchers employed, among other things, an n-gram model to identify and remove promotional tweets from their data-set. This helped to achieve a prediction accuracy of 83%.

2.2 The best models for crypto-currency predictions

Recently, Pagolu et al.[SOURCE]'s work has shown that LSTM's could be outdone by an autoregressive-recurrent neural network on temporal prediction tasks. They achieved 59% accuracy through solely analyzing previous behavior.

These findings heavily informed my approach to cleansing the tweet data-set. Additionally, they provided benchmark metrics that I could compare with my algorithm's performance.

3 Dataset and Features

The dataset consists of 100k tweets from the Twitter Search API and a small (still incomplete) list of crypto-currency related news articles from the Google News API, all stored on AWS' S3. These are analyzed in conjunction with 2.4M price change data points, across 100 different currencies, accessed through the CryptoCompare API. I also downloaded the historic minute by minute bitcoin dataset to train on. In order to optimize performance, given that the CryptoCompare API was limited to the past 36 hours, data was split into 95% train, and 5% for dev and test.

Tweets underwent the following pre-processing and normalization steps:

1. Remove promotional, spam and non-nonsensical tweets using a naive bayes classifier
2. Remove repeated tweets
3. Create an influence score for each tweet, derived from likes, favorites and replies
4. Run sentiment analysis on tweets using Stanford's "Deeply Moving" algorithm that's integrated into Stanford CoreNLP

5. Quantify each tweet by multiplying its sentiment by its influence score
6. Combine tweets into discrete, minute long, time periods and sum each tweet's score
7. Normalize the dataset around 0

Currency price data underwent the following pre-processing and normalization steps:

1. Transform the price vector into a price change vector
2. Create a matrix of price change vectors that indicate the change in price over five different time periods
3. Normalize raw change into percent change

4 Methods

4.1 The Two Model Structure

My algorithm executes predictions through the use of two separate LSTM-RNN models. The first model is deep, with large input vectors of aggregated data. This model is designed to produce general insights about crypto-currency price fluctuation. It allows currencies to be trained on a uniquely large dataset. This dataset has 3.3M data-points, and is constantly growing. This massive dataset provides the freedom of using a deep-RNN. The next model is initialized with the weights of the prior one, but is optimized for each currency. Only that specific currency's data is fed into the model. This approach combines the benefits of big data and individual customization while building a model.

To test the effectiveness of my transfer-learning approach I compared the exact same algorithm run with the two different initialization approaches. The standard transfer of weights approach improved net accuracy by 1.5%. Below you can compare the two different initializations:

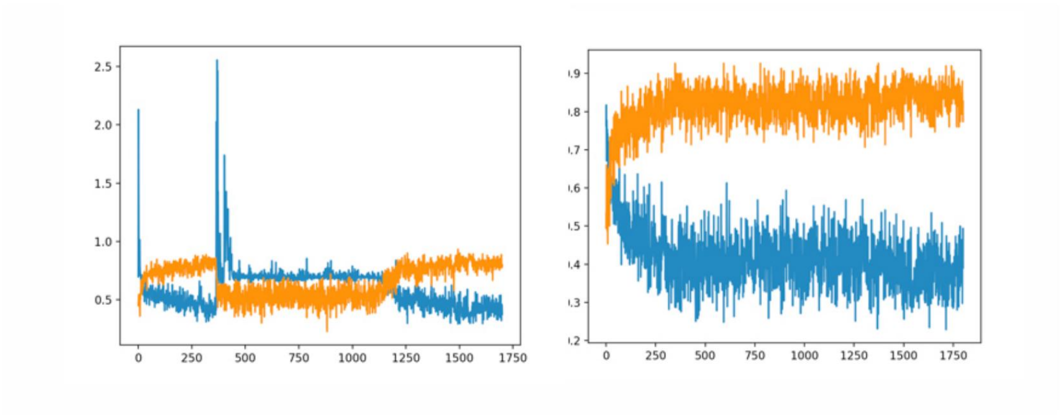

Figure 1: Random Initialization (Left), Transfer Learning (Right)

4.2 Overview

Both RNN models use mini-batch gradient descent and forward propagate with a multi-layer dynamic-RNN and three LSTM layers. The algorithm takes the first input in the batch, processes it through each hidden layer, and then recurrently repeats the process with the previous output state and the next input. The final iteration produces a predicted return for each input set. My algorithm then uses truncated backwards propagation to get the weight's gradients.

A softmax layer is applied to the saved prediction set. Next, the total loss is computed using cross entropy loss. After the loss is calculated, the Adam Optimizer[SOURCE] is applied to train the model.

$$Loss = -\frac{1}{n}\sum_{i}^{n} Y\ln(X) + (1-Y)\ln(1-X)$$

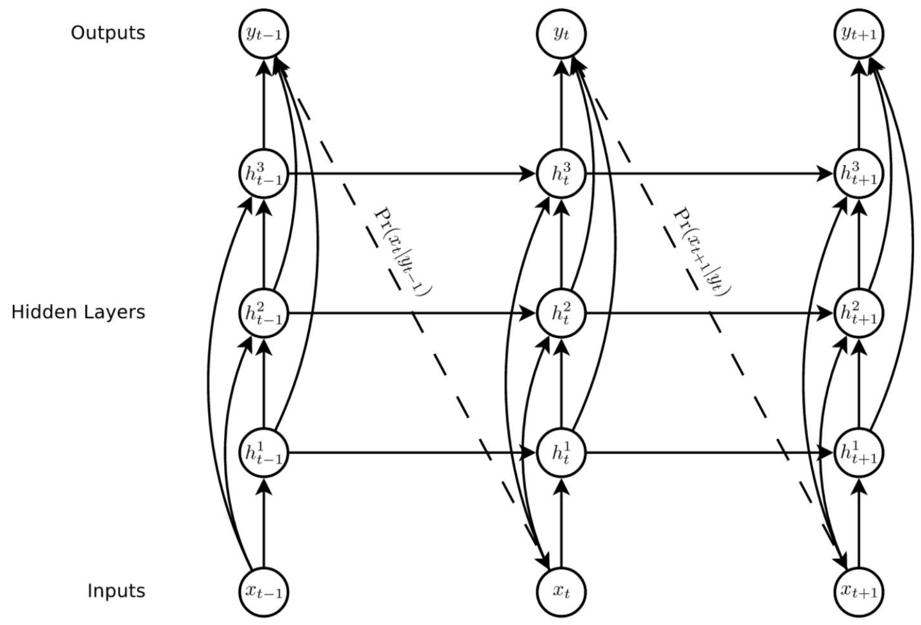

Figure 2: An illustration of a RNN with three LSTM cells

5 Experiments/Results/Discussion

5.1 Intro

Experiments for this project were two fold. First, the main RNN had to be optimized with low variance to act as a good weights initialization matrix. Next, each currency had a model that needed to be trained and optimized. I focused on optimizing hyperparameters for Bitcoin, because I suspected that Bitcoin would have the most universal hyperparameters.

5.2 Early Training

Early training was very frustrating because I accidentally set the size of the LSTM cells too small by a factor of 100. This made training confusingly ineffective. I optimized everything I could but it didn't help. Then, I realized that the states size was optimized at 256, as opposed to my earlier 4. Luckily, the optimizations that previously had very little effect, now greatly improved the accuracy.

Training had another challenge in that training any given model could take many hours. Time-per-epoch could differ massively between two models that could deliver similar qualities. I was forced to choose one of two algorithms, one which returns higher and faster results in the short term, or an algorithm which might out perform the other model in the long run.

5.3 Results

Prediction Results	**Evaluation Metrics**		
	Training%	**Test%**	**Normalized%**
Bitcoin 1 Minute	76.41	54.13	50.23
Bitcoin 5 Minutes	83.32	57.34	54.01
Bitcoin 15 Minutes	82.24	55.34	50.67
Bitcoin 30 Minutes	79.2	55.06	52.24
Bitcoin 1 Hour	82.42	59.35	53.21

Hyper-Parameters: Currently, the algorithm has a truncated backprop length of 30, a state size of 128, three layers, a beta1 of .9, a beta2 of .999 and a learning rate of .01. I believe these hyper-parameters can be optimized further. My algorithm first groups the data into five batches, with a mini batch size of 30 in each batch.

6 Conclusion/Future Work

Given the limited time frame of data that my model trained on, I would not recommend using it for actual trading, nor would I declare my results scientifically conclusive. There are a number of improvements I want to make on the model. For example, I plan on implementing a customized loss function to penalize over-estimates more than under-estimates. This would have the effect of predicting negative change really well, while predicting positive change with high precision, yet low frequency. Additionally, I am experimenting with my model as a multi-class classifier and am considering switching in order to potentially yield more precise predictions.

My model achieved its highest performance predicting the likelihood of price change five minutes from a given time period.

7 Contributions

Aaron Moss provided guidance and help in implementing the twitter API.

I have always struggled with the implementation of algorithms as opposed to the concepts. In order to help address my weaknesses, I wanted to do this project alone because I knew I would learn a lot more.

References

[1] Andrew Jonathan Slottje. Ian Edward Shaw. Samuel Karl Joel Persson (2018) Hybrid Autoregressive-Recurrent Neural Network Architecture for Algorithmic Trading of Cryptocurrencies. Stanford CS 230.

[2] Abdulmalik Mahmoud Obaid. Dante Zakhidov. (2018) Predicting Cryptocurrencies Fluctuation based on Public Opinion

[3] Connor Lamon. Eric Nielsen. Eric Redondo. (2018) Cryptocurrency Price Change Prediction based on News and Social Media Sentiment. Stanford CS 230.

[4] Venkata Sasank Pagolu. Kamal Nayan Reddy Challa. Ganapati Panda. Babita Majhi (2016) Sentiment Analysis of Twitter Data for Predicting Stock Market Movements. Stanford CS 230. https://arxiv.org/abs/1610.09225

[5] EVITA STENQVIST. JACOB LÖNNÖ. (2017) Predicting Bitcoin price fluctuation with Twitter sentiment analysis https://kth.diva-portal.org/smash/get/diva2:1110776/FULLTEXT01.pdf

[6] Hong Kee Sul. Alan R. Dennis. Lingyao (Ivy) YuanTrading on Twitter: (2016) Using Social Media Sentiment to Predict Stock Returns https://onlinelibrary.wiley.com/doi/full/10.1111/deci.12229

[7] Stuart Colianni. Stephanie Rosales. Michael Signorotti. (2015) Algorithmic Trading of Cryptocurrencies Based on Twitter Sentiment Analysis

[8] https://github.com/muatik/naive-bayes-classifier

[9] Diederik P. Kingma & Jimmy Lei Ba (2014) ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION