Eth Lab: Predicting the Price of Ether Using RNNs

Michael Zhu
mzhu@cs.stanford.edu

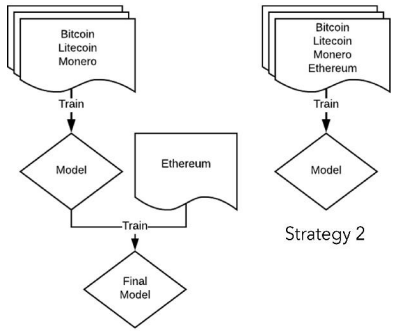
CS 230
Stanford University
Winter 2018

Introduction

- Cryptocurrencies are volatile
- Would be nice to be able to predict their prices
- Why not use deep learning?
- Challenges: not much data, lots of noise

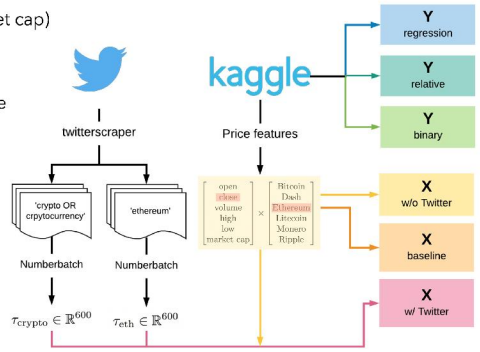
Data Augmentation

- Insight 1: Ethereum has only been around for a couple of years, but other cryptocurrencies may have similar behavior
- Obtain feature vectors as described in Data & Features section, but for Bitcoin, Litecoin, and Monero instead of Ethereum
- Only use data from last 300 days of dataset (since dev/test set come from last 100 days)
- 2 strategies:
  - Train on {Bitcoin, Litecoin, Monero}, use params with best validation error to warm start training on Ethereum
  - Shuffle all of {Bitcoin, Litecoin, Monero} into Ethereum training data
- Second strategy more effective



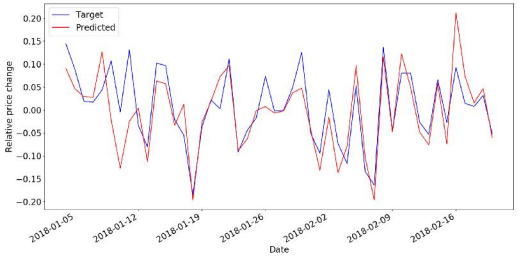
Data & Features

- Kaggle cryptocurrency dataset for price, price-related features (e.g. high, low, market cap)
- Scraped Twitter for news tweets matching queries 'crypto OR crpytocurrency' and 'ethereum' for every day since start of Ethereum
- Compute Numberbatch embedding (dim 300) for each word in corpus, concatenate element-wise max and min to obtain (dim 600) vector for corpus
- 3 prediction tasks:
  - Regression: predict next day's closing price for ethereum
  - Relative: predict relative change (e.g. -3% from today's price)
  - Binary: will the price go up or down?
- 3 feature vector variants (window sizes 5 and 10):
  - With Twitter: Twitter and price
  - Without Twitter: only price features
  - Baseline: only previous closing prices of Ethereum



Results & Analysis

- Final model: 2 LSTM layers and 1 linear layer with 200 hidden units each, followed by linear output layer
- Loss function: Mean squared error for regression and relative prediction tasks, binary cross-entropy for binary prediction task
- Twitter features did not help (too high-dimensional?)
- Model performs well on the relative prediction task, poorly on regression
- Model trained on relative prediction can be used for regression, outperforms model trained on regression task
- Baseline models:
  - Above model, trained on "baseline" features (all tasks)
  - Lag model: predict label to be same as previous day (all tasks)
  - Lag* model: predict label to be the average of the last 2 days (regression, relative)
  - All 0: always predict relative change to be 0 (relative)
  - Majority: Predict label to be the mode of the last 3 labels (binary)



	Mean Squared Error
Best model	0.13365
Same model, no data augmentation	0.28236
Baseline NN (window size 10)	0.32275
Lag* baseline	0.46616
Always 0 baseline	0.30834

Test results for "relative" prediction task