CSE 515T Final Report: Bayesian Regression for Crypto Currency Prediction

Wenzhen Zhu (445518)

Introduction

I started buying BTC since 2015 summer. I earned a lot in 2017 (without really doing anything but buying and building portfolios) and lost a lot of the profit since 2018 Jan. I really enjoyed gambling. After a huge loss in 2018 April, I started to read papers in finance and try to learn how to gamble smartly. Getting passive income is my primary motivation for doing this project, because I don't want to spend my whole life doing software development for some company. I want to have freedom to pursue things I really like. So I plan to keep working on it after graduation since I would have more time by then. Since the orderbook of Cryptocurrency on GDAX is really similar to stocks on Robinhood, I am also planning to start trading stocks once I receive my share from company.

This project is inspired by a paper `Bayesian regression and Bitcoin`.

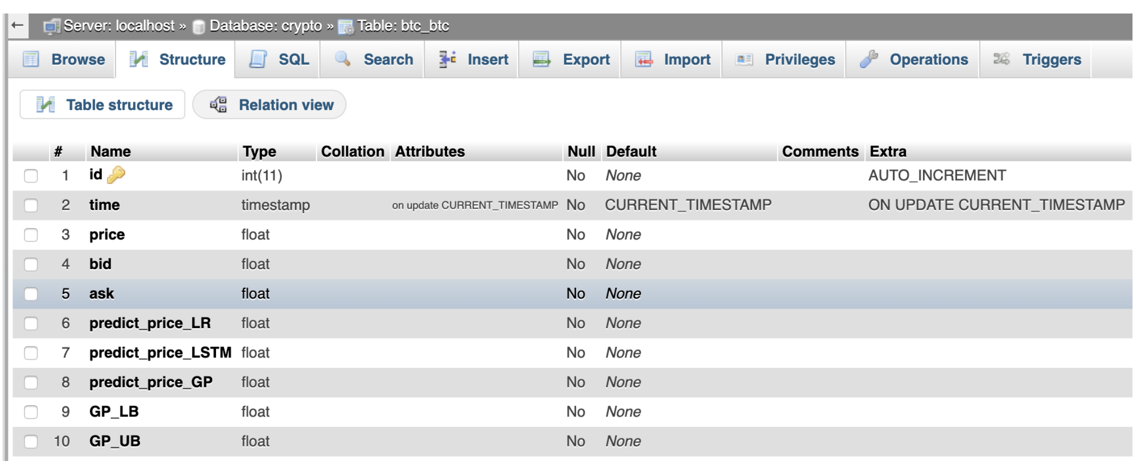
Project Design

```
CryptoTrading
├── backend/
│   ├── gdaxCrawler50.py: crawler that collect data for training
│   ├── publicClient.py: client connect with Coinbase API
│   └── writeToDB.py: write testing data to dabase
├── crypto/
│   ├── btc
│   ├── crypto
│   ├── manage.py
│   └── db.sqlite3
├── data/
│   └── btc.csv: training data (1.47GB) emitted because the file is too large
├── model/
│   ├── GP.wxf: smaller GP model trained with first 5000 datapoints (~ 200 MB)
│   ├── GP_big.wxf: bigger and better GP model with 10,000 datapoints (~ 800 MB)
│   ├── p1.wxf: predict price change deltaP1 based on previous 30 mins data
│   ├── p2.wxf: predict price change deltaP2 based on previous 60 mins data
│   ├── p3.wxf: predict price change deltaP2 based on previous 120 mins data
│   └── pFinal.wxf: bayesian regression model with input (p1, p2, p3, gamma)
├── test/
│   ├── lr_test.wl
│   └── gp_test.wl
└── train/: contains model training scripts
    ├── GaussianProcess.nb
    └── BayesianRegression.nb
```

Project Details

Database Setup

Since we are mainly focusing on 4 features: time, price, bid volume, ask volume, we firstly need to craw the order-book on GDAX to obtain such 4 features to train / test our model. After there are 720 data-points in database, the testing script can be ran to compute next 10 second price prediction. There were 3 models I planned to do, Bayesian Linear Regression, LSTM, Gaussian Process, hence column 6-10 are for testing script to write into database and visualization website is updating in real time.



#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	**id**	int(11)			No	*None*		AUTO_INCREMENT
2	**time**	timestamp		on update CURRENT_TIMESTAMP	No	CURRENT_TIMESTAMP		ON UPDATE CURRENT_TIMESTAMP
3	**price**	float			No	*None*		
4	**bid**	float			No	*None*		
5	**ask**	float			No	*None*		
6	**predict_price_LR**	float			No	*None*		
7	**predict_price_LSTM**	float			No	*None*		
8	**predict_price_GP**	float			No	*None*		
9	**GP_LB**	float			No	*None*		
10	**GP_UB**	float			No	*None*		

To write real-time data into database, run the following command in root directory of this project:

```
python3 backend/writeToDB.py
```

Here is an example of the data collected in MySQL database:

id	time	price	bid	ask
32786	2019-04-27 10:57:50	5158.13	102.196	335.91
32787	2019-04-27 10:58:01	5158.02	113.878	84.2533
32788	2019-04-27 10:58:11	5158.02	109.444	94.4806
32789	2019-04-27 10:58:21	5158.02	108.964	95.2198
32790	2019-04-27 10:58:31	5158.02	109.912	827.065
32791	2019-04-27 10:58:41	5158.02	103.597	1416.5
32792	2019-04-27 10:58:52	5158.02	100.243	615.865
32793	2019-04-27 10:59:02	5157.58	126.247	356.597
32794	2019-04-27 10:59:12	5157.58	101.633	362.144
32795	2019-04-27 10:59:22	5156.65	103.064	260.591
32796	2019-04-27 10:59:32	5156.9	106.71	251.45
32797	2019-04-27 10:59:42	5156.9	107.791	253.127
32798	2019-04-27 10:59:53	5156.82	107.174	247.411
32799	2019-04-27 11:00:03	5156.82	98.1742	779.541
32800	2019-04-27 11:00:13	5155.72	155.431	160.86
32801	2019-04-27 11:00:23	5155.73	143.472	98.9053
32802	2019-04-27 11:00:33	5156.83	140.83	89.9579
32803	2019-04-27 11:00:43	5156.84	141.171	98.0376
32804	2019-04-27 11:00:54	5156.84	142.172	91.8121
32805	2019-04-27 11:01:04	5156.84	186.675	93.0959

Model Evaluation

To run Bayesian Regression / Gaussian Process model evaluation, type the following command:

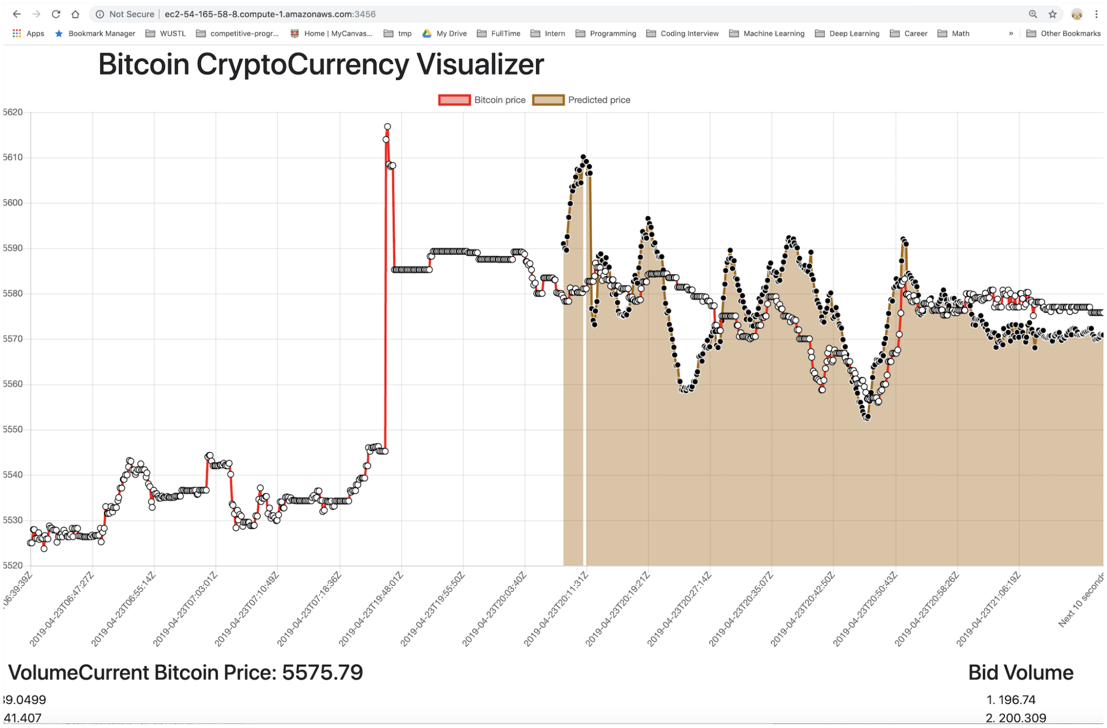
```
wolframscript -f test/lr_test.wl
wolframscript -f test/gp_model_test.wl
```

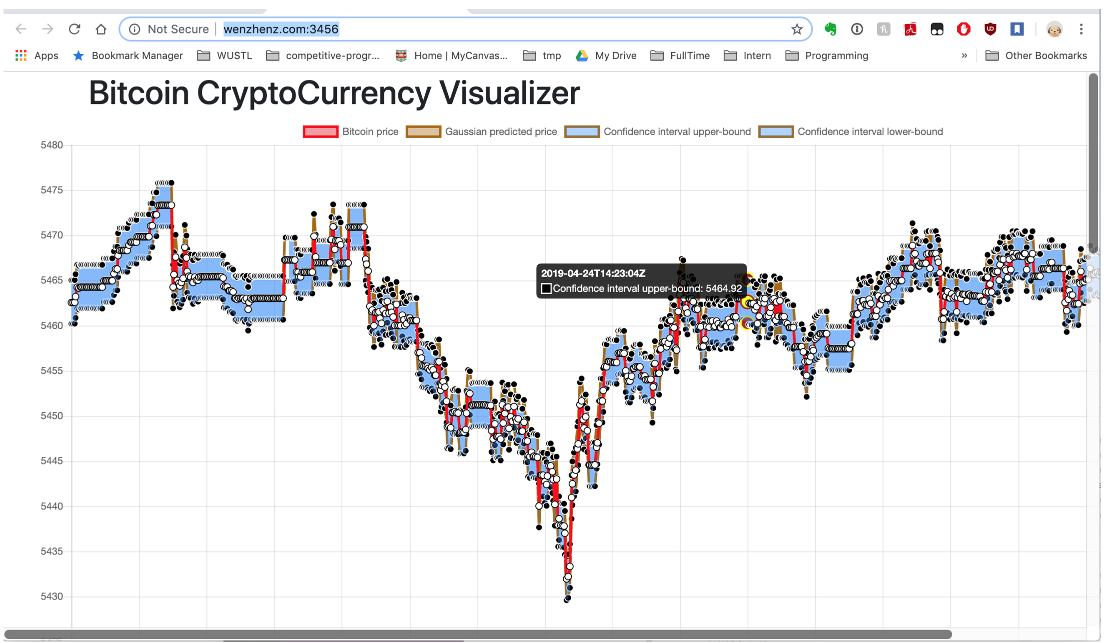
Front End

Usage script:

```
python crypto/manage.py runsedver 0.0.0.0:3456
```

Front end uses Django framework to display a chart of real-time Bitcoin price as well as forecasting next 10 second predicted price.





I spent lot of time to make the real-time testing happens because in the future, I want to explore the online-training algorithms, and I heard from some Quant friend in HedgeFound trading company that sometimes model only works for a short time period, a real-time testing platform could reflect the model’s performance in real-time.

Back end

Backend is split into 2 parts:

- Model
  - Crawler to collect data (timestamp, price, bid_volume, ask_volume) every 10s for training purpose.
  - Implemented a Bayesian Regression model proposed on paper.
  - Trained several models
    - Bayesian Regression model
    - Gaussian Process model
  - Python script that query the real-time data and evaluate on pre-trained models
- DJANGO Framework
  - Takes information from the database, presented as a model in the Django framework and sends it to the user trying to access this website.

3. Model

To predict label y given associated observation x, we can utilize the conditional distribution of y given x given as follows:

$$P(y \mid x) = \sum_{k=1}^{T} P(y \mid x,\ T=k)\, P(T=k \mid x)$$

$$= \sum_{k=1}^{T} P(y \mid x,\ T=k)\, P(x \mid T=k)\ P(T=k)$$

$$= \sum_{k=1}^{T} P_k(y)\, P(\epsilon = (x - s_k))\, \mu_k =$$

$$\sum_{k=1}^{T} P_k(y)\, \exp\left(\frac{-1}{2}\, \|x - s_k\|_2^2\right) \mu_k \ (3)$$

We want to utilize empirical data as proxy for estimating conditional distribution of y given x given in (3), Specifically, given n data points $(x_i,\ y_i)$, 1 <= i <= n, the empirical conditional probability is:

$$P_{\text{emp}}(y \mid x) = \frac{\sum_{i=1}^{n} \mathbb{I}(y=y_i)\, \exp\left(\frac{-1}{4}\, \|x-x_i\|^2\right)}{\sum_{i=1}^{n} \exp\left(\frac{-1}{4}\, \|x-x_i\|^2\right)} \ (4)$$

Estimation of the conditional expectation of y, given observation x:

$$E_{\text{emp}}(y \mid x) = \frac{\sum_{i=1}^{n} y_i\, \exp\left(\frac{-1}{4}\, \|x-x_i\|^2\right)}{\sum_{i=1}^{n} \exp\left(\frac{-1}{4}\, \|x-x_i\|^2\right)} \ (6)$$

This estimation can be viewed equivalently as a linear estimator, let vector X(z) $\in \mathbb{R}^n$ be such that $X(z)_\alpha = \exp\left(\frac{-1}{4}\, \|z - x_\alpha\|^2\right)$ /Z(x) with $Z(x) = \exp\left(\frac{-1}{4}\, \|z - x_\alpha\|^2\right)$, and $y \in \mathbb{R}^n$ with i-th component being y_α then $\hat{y} = E_{\text{emp}}[y \mid z]$ is

$$\hat{y} = X_\alpha(z)\, y_\alpha \ (7)$$

(7) is used for predicting future variation in the price of Bitcoin.

Predicting Price change:

Quote from paper:

The core method for price change Δp over the 10 second interval is the Bayesian regression in (7). Given time- series of price variation of Bitcoin over the interval of few months, measured every 10 second interval, we have a very large time-series (or a vector).

We use this historic time series and from it, generate three subsets of time-series data of three different lengths: S_1 of time-length 30 minutes, S_2 of time-length 60 minutes, and S_3 of time-length 120 minutes.

Now at a given point of time, to predict the future change Δp, we use the historical data of three length: previous 30 minutes, 60 minutes and 120 minutes - denote x_1, x_2 and x_3. We use x_j with historical samples S_j for Bayesian regression (as in (7)) to predict average price change for We also calculate where v_{bid} is total volume people are willing to buy in the top 60 orders and v_{ask} is the total volume people are willing to sell in the top 50 orders based on the current order book data. The final estimation Δp is produced as

$$\Delta p = w_0 + \sum_{j=1}^{3} w_j \Delta p_j + w_4 \gamma$$

where $w = (w_0, \ldots, w_4)$ are learnt parameters. In what follows, we explain how S_j $(1 \le j \le 3)$ are collected; and how w is learnt. This will complete the description of the price change prediction algorithm as well as trading strategy.

In the beginning, I used the method the paper described, but I made a mistake: I am supposed to to use $\left(\vec{S_j}, \Delta p\right)$ as the training, but I forgot to calculate price change Δp and used price p instead. Therefore, in the beginning I was not very confident of this model

I have 79582 datapoints in total, and learning parameters fo Δp_j takes a long time, so I stored fundamental models Δp_j as .wxf file in CryptoTrading/data/, such as dp1Data.mx, (because .wxf saves symbols and .mx saves only data). Then we can just export them easily in next steps.

From here I made two different approaches:

- 1. Follow the paper and use linear regression model to learn the final price change.
- 2. Take first 5000 datapoints and 10,000 datapoints to train a Gaussian Process model.
  - We discussed in class, when we have a lot of data, we should not use GP.
  - Training a GP model with same dataset on my laptop (16GP memory) is very difficult, and kernel crashed many times during training, therefore, I reduced the data from 80k to 5k and 10k. I didn't have enough time to finish training, hence I halted the training process in the middle. And the GP model trained is ~800MB, which I cannot evaluate on my Amazon EC2 instance, because my instance type is t2.micro, which only has 1G memory. While the smaller GP model I trained is not good enough to demonstrate.

Model Efficiency

This paper didn't define any metrics to evaluate a model's efficiency, it only says "in 4 months, they doubled their capital within 2 months", therefore, I wanted to make a website that can show how good a model is in real time.

For Bayesian Linear regression, I believe this model is reasonable. For example, after collecting 2 hours / 720 datapoints, the testing script gets stable and shows that this model can predict the market's important behavior, such as a sudden increase and sudden decrease before such increase / decrease happens.