Feature Extraction and Prediction on Bitcoin Data

Oscar Bergqvist, Martin Studer, Cyril de Lavergne

October 2019

Abstract

The goal of this report is to explore various techniques for the prediction of the increment of the price of bitcoin. Since the raw data is high dimensional, a key challenge is the extraction of suitable features. We use GAN's, Variational Autoencoders, and Signatures of Rough Paths to obtain features. Using those features, we use Lasso, Dilated CNNs, and LSTMs to predict the future bitcoin price. The Features obtained from the Variational Autoencoder combined with the LSTM gave the best result. Features from Signatures of Rough Paths combined with Lasso also performed decently. In section 7 we summarize our findings.

1 Introduction

We use minute-level Bitcoin data. Various quantities such closing price, highest price or volume are recorded. Our goal is to predict the sign of the increment of the close price during the next minute. Hereby we restrict ourselves to situations where the sign of the increment is in $\{+1, -1\}$ ignoring the situation when the increment is 0. This is reasonable from a financial perspective, because if the increment is 0, we're happy with either prediction.

In order to reduce the dimensionality of the data we perform feature extraction. This is done with Signatures of Rough Paths, Variational Autoencoders, and GANs. Afterwards we introduce suitable prediction methods to exploit the obtained features and record the results we obtain.

The reminder of the report is organised as follows: In section 2 we discuss how we extracted features using a GAN. In section 3 we perform prediction on using a Dilated CNN on both the raw data and the features extracted by the GAN. In section 3 we discuss how we extracted features using Signatures of Rough Paths. Since prediction is performed by a simple logistic Lasso, we directly incorporate prediction into this section. In section 5 we we how we extracted features using a VAE. In section 6 we perform prediction on the features obtained from the VAE using a LSTM neural network.

2 Feature Extraction using GAN

The Generative Adversarial Network (GAN) was introduced by Goodfellow et. al. in the 2014 paper *Generative Adversarial Nets* [3]. GAN operates through the simultaneous training of two models: a generative model $G(\mathbf{z})$ that tries to produce samples from the data distribution (using the latent variable $\mathbf{z}$) and a discriminative model $D(\mathbf{x})$ that tries to distinguish between the true data distribution and samples from G. Learning is achieved by minimizing the value function

$$\min_G \max_D \mathbb{E}[\log(D(\mathbf{x}))] + \mathbb{E}[\log(1 - D(G(\mathbf{z})))]. \tag{1}$$

In practice training is preformed by interchangeably optimizing the discriminator and the generator. The training stops when both the generator and discriminator has converged to have 50% accuracy.

2.1 Wasserstein GAN and Improved Wasserstein GAN

In practice, the implementation in [3] has been proven to suffer from lack of robustness for changes in model architecture. Since 2014 several other algorithms have been proposed, such as the *Wasserstein GAN* (WGAN) [2], and improved Wasserstein GAN (WGAN-GP) [4].

In the Wasserstein GAN the value function is

$$\min_G \max_D \mathbb{E}[D(\mathbf{x})] - \mathbb{E}[D(G(\mathbf{z}))] \tag{2}$$

The WGAN value function is correlated with the quality of samples, which is not the case for GAN. Also, the gradients for the WGAN are more well behaved then the ones for GAN, which makes traning of WGAN easier.

In WGAN-GP the value function is

$$\min_G \max_D \mathbb{E}[D(\mathbf{x})] - \mathbb{E}[D(G(\mathbf{z}))] + \lambda \mathbb{E}[(\|\nabla D(\hat{\mathbf{x}})\|_2 - 1)^2] \tag{3}$$

where *lambda* is a gradient penalty parameter and $\hat{\mathbf{x}}$ is sampled uniformly on the line between the real sample $\mathbf{x}$ and the generated sample $\tilde{\mathbf{x}} = G(\mathbf{z})$. The authors found $\lambda = 10$ to be a good value across several different model architectures. Another difference from GAN is that no batch normalization is needed.

2.2 Training of Improved WGAN for Feature Extraction

The WGAN-GP was implemented using neural networks for both the discriminator and generator. The aim for the generator is to generate one hour of the nine dimensional bitcoin data, which can be represented by a $60 \cdot 9 = 540$ dimensional vector. This will thus be the output dimension of the generator and the input dimension of the discriminator. The data is aggregated so that one sample contains the time series data from $t = 1$ to $t = 60$ and the next sample contains the data from $t = 2$ to $t = 61$ and so on.

The generator takes a 100 dimensional latent vector as input, and consists of three fully connected layers and ReLu activation between each intermediate layer. The last layer is linear as described in the previous section. L1 and L2 regularization is applied on each layer to prevent the weights from getting too big. The discriminator consists of two fully connected layers. Regularization and ReLu activation is used for the first layer, and the output is a scalar value. The aim is to create a generator which is more powerful then the discriminator, since learning the data distribution can be seen as a more difficult task then prediction.

Training was performed using the Adam optimizer with a batch size of 10 and learning rate 0.0001, $\beta_1 = 0.5$ and $\beta_2 = 0.9$. The resulting loss functions of the generator and the discriminator are presented in Figure 1.

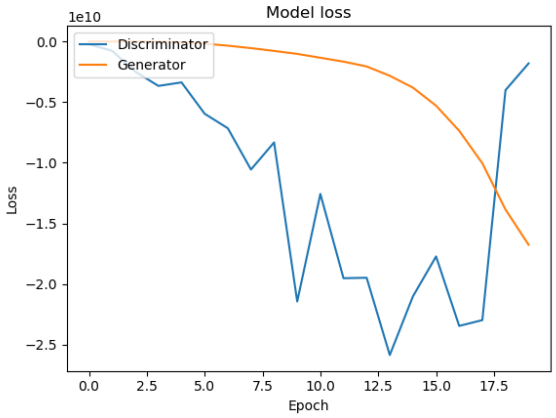


Figure 1: The generator and discriminator loss when training GAN for 19 epochs.

After training the GAN the output of the next last layer of the discriminator is extracted as features for prediction. This corresponds to a 10-dimensional vector for each 60 minutes of bitcoin data.

3 Prediction using Dilated Convolutional Neural Networks (Dilated CNNs)

Dilated CNNs are powerful modification of CNNs for time series data. For example DeepMind's WaveNet [7] is able generate speech which mimics any human voice and sounds more natural than the best existing Text-to-Speech systems. In our project we used Dilated CNNs to predict Bitcoin prices.

3.1 Dilated CNNs

A Dilated CNN is a CNN which is tailored to time series data. It's main feature is that it's debt only increases with the logarithm of the number of time steps we incorporate into the model. This makes it suitable to capture long term dependencies with relatively few parameters. Figure 2 gives a good illustration of a Dilated CNNs.

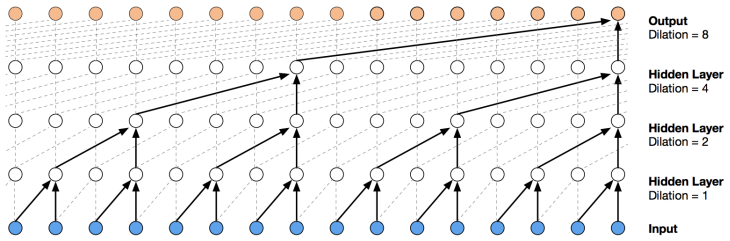


Figure 2: Dilated CNN architecture, image form [7]

Compared to the classical CNN the main difference is that instead of doing a convolution between two cells next to each other, we make a convolution between two cells which are 2^{l-1} apart, where l is the number of the hidden layer. This allows us to keep the number of hidden layers low, while still incorporating a high amount of information. As mentioned before the required debt for this architecture increases with the logarithm of the number of time steps we incorporate. Figure 3 is the corresponding illustration of a classical CNN. It gives a good understanding why Diluted CNNs are better suited for capturing long term dependencies.

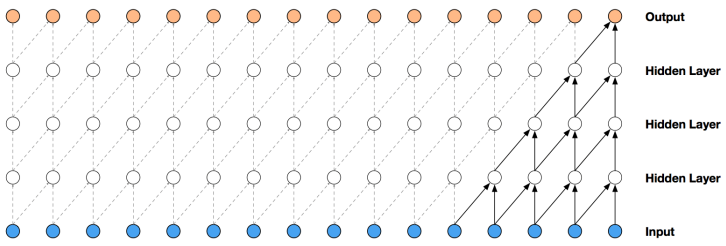


Figure 3: Classical CNN architecture, image form [7]

3.2 Our approach

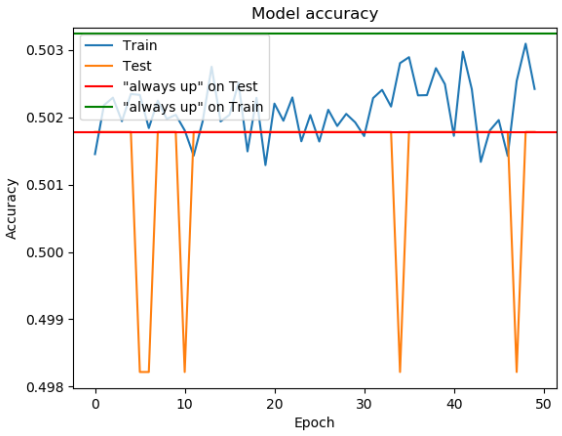
We use the data from the last 64 time intervals to predict the sign of the next increment. To handle 64 time intervals we need exactly $6 = \log_2(64)$ convolutional layers. Hereby we decrease number of filters from 20 to 11 to 7 and use L2-regularisation. Finally we train a fully connected ReLU hidden layer with 80 nodes and dropout regularisation, followed by the map to the output. We train this architecture with the Adam optimiser (with default parameters) over 50 epochs. This approach is inspired by [1].

3.3 Results on raw data

Our network converged very quickly. However the result is disappointing. We obtain a prediction accuracy of 50.2%. This is similarly accurate to always predicting to most probable class. From our initial approach we have done the following modifications hoping to improve the accuracy

1. replace ReLU with leaky ReLU activation
2. try different degrees of l2 and l1 regularisation
3. change the number of convolutions

However, all models behaved similarly to the final one.



3.4 Results on GAN features

The resulting accuracy using the GAN features is presented in Figure 4.

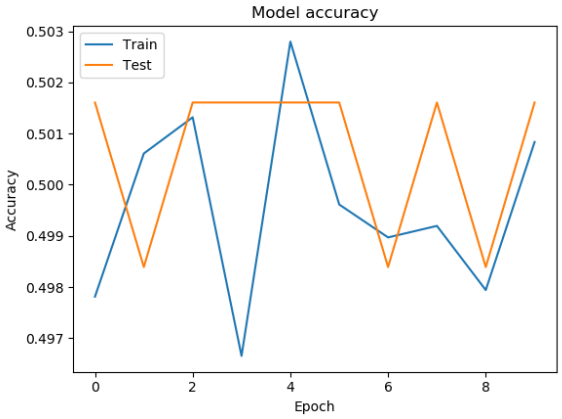


Figure 4: The training and test accuracies for prediction using the GAN features.

It is clear that using these features does not give a better accuracy then random guessing, which could indicate a failure of the GAN to capture a lot of information about the time series structure in the features.

4 Feature extraction using Signatures of rough paths and prediction with Lasso

Originally developed to solve problems in partial differential equations and stochastic calculus, signatures of rough paths have become useful way to extract features for machine learning. For example Yang, W. et. al. [8] has been able to obtain state of the art results in the classification of handwritten Chinese letters using features based on signatures of rough paths. Our goal is to use signatures of rough paths to predict bitcoin prices.
As it turns out signatures of paths are enough for the purposes of our task. Thus instead of discussing signatures of rough paths we'll just discuss signatures of paths.

4.1 Definition of the signatures of a path

Let $X : [0, T] \ni t \mapsto X_t = (X_t^1, ..X_t^d) \in \mathbb{R}^d$ be a continuous path of bounded variation. Let $\mathcal{I}$ be the set of all multi-indexes $(i_1, ..., i_k)$ with $k \in \mathbb{N}_0$, $i_1, ..., i_k \in \{1, ..., d\}$. Then for all $0 \leq s \leq t \leq T$, $I \in \mathcal{I}$ we define

$$X_{s,t}^I = \int_{s \leq u_1 \leq u_2 ... \leq u_k \leq t} dX_{u_1}^{i_1} ... dX_{u_k}^{i_k} \tag{4}$$

The set $(X_{s,t}^I)_{I \in \mathcal{I}}$ is called the signatures of the path X on $[s, t]$

4.2 Key properties of signatures of a paths

Here we list some properties of signatures of paths to give reader a basic understanding of the approach. For details we refer to [6].

1. **Uniqueness** The set $(X_{s,t}^I)_{I \in \mathcal{I}}$ uniquely (to be precise up to tree like equivalence) determines the function $[s, t] \ni u \mapsto (X_u - X_s) \in \mathbb{R}^d$.

2. **Linearity** For all $I, J \in \mathcal{I}$ there exist $\mathcal{I}_{I,J} \subset \mathcal{I}$ such that

$$X_{s,t}^I X_{s,t}^J = \Sigma_{K \in \mathcal{I}_{I,J}} X_{s,t}^K \tag{5}$$

3. **Number of signatures up to an order** Let d be the dimension of the path, let $n \in \mathbb{N}$, and let $\mathcal{I}_n = \{I = (i_1, .., i_k) \in \mathcal{I} \colon \sum_{l=1}^{k} i_l \leq n\}$. Then

$$|\mathcal{I}_n| = \frac{d(d^n - 1)}{d - 1} \tag{6}$$

We hence say that there are $|\mathcal{I}_n|$ signatures up to order n

4. **The most in important information is in the low order terms** Making this mathematically precise a is a beyond the scope of this report. But restricting ourselves to low order terms is theoretically justified.

4.3 General approach

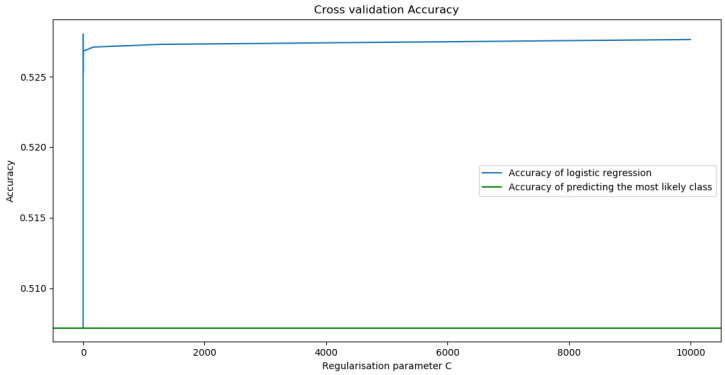
The standard approach (for example discussed in [5]) for prediction using signatures of paths is to first compute the signatures up an order between 3 and 6. Ignoring higher order terms is justified since property 4 implies that most relevant information is stored in the low order term. Then one uses linear models to do the prediction. This seems a bit counter intuitive at first, because the transform is nonlinear. However property 2 above implies that we can represent polynomials of signatures as a linear sum of signatures of different order. Since polynomials can approximate continuous functions by Stone–Weierstrass theorem, using just linear models is justified. Furthermore it's worth noting that the number of features grows very fast in the dimension d of the path. Hence training anything else than a linear model is not feasible. The most commonly used linear model for prediction labels in such a setting is logistic Lasso. This method of prediction performs well in high noise settings and does a categorical prediction. Normalising the predictors is a standard procedure in any Lasso regression and therefore also used here.

4.4 Approach in our setting

First it's worth noting that our data has missing numbers. For simplicity we fill those in by forward filling. We could create indicator variables to indicate if this is an original or a filled value. Such a model would be more precise, but make our biggest problem, the already large number of predictors even bigger. Our raw data is 45 dimensional. Extracting features for such a high dimensional path is not feasible. Therefore we choose the 4 predictors we consider most relevant and build our model on them. Furthermore we restrict ourselves to signatures up to order 3. This yields 84 signatures, which is already barely commutable on a personal computer. For each prediction we calculate the signatures of the path of the last 500 ticks and predict the sign of the returns 60 ticks later. This means that the amount of time we look forward and backwards varies. However since our path has a time coordinate this information is incorporated into the prediction.

4.5 Results

The best model after crossvalidation obtains a an average accuracy score of 52.8%. The model predicting the most likely class is only able to get an accuracy 50.7%. Therefore our model is able to detect some information inside the signatures. Furthermore 25 out of 84 predictors are used in the best model. To get a a full understanding of the strength of models based on signatures of rough paths we would need to train without blind forward filling, calculate signatures of higher order, and use the entire set of predictors. Furthermore, since we implicitly turn discrete data into a continuous path various interpolation strategies (rather than the standard interpolation provided by the library) could be explored. Properly training such a model would need a supercomputer.



5 Feature extraction using a Variational Autoencoder

Variational Autoencoder (VAE) is rooted from bayesian inference. In fact, it determines the underlying probability distribution of data so that it sample new data from that distribution over a given number of latent variables.
The relation of data with the latent variables is as follows:

$$P(X) = \int P(X|z)P(z)dz$$

The goal of the variational autoencoder is given a metric to compare how different are actual distribution of data compared to easier distributions such as Gaussian. One metric often used is Kullback–Leibler divergence with Q denoting a simple distribution.

$$\begin{aligned} D_{KL}[Q(z|X)\|P(z|X)] &= \sum_z Q(z|X)\ \log \frac{Q(z|X)}{P(z|X)} \\ &= E\left[\log \frac{Q(z|X)}{P(z|X)}\right] \\ &= E[\log Q(z|X) - \log P(z|X)] \end{aligned}$$

From Bayes' rule

$$\begin{aligned} D_{KL}[Q(z|X)\|P(z|X)] &= E\left[\log Q(z|X) - \log \frac{P(X|z)P(z)}{P(X)}\right] \\ &= E[\log Q(z|X) - (\log P(X|z) + \log P(z) - \log P(X))] \\ &= E[\log Q(z|X) - \log P(X|z) - \log P(z) + \log P(X)] \end{aligned}$$

We can hence move the equation as follows:

$$D_{KL}[Q(z|X)\|P(z|X)] - \log P(X) = E[\log Q(z|X) - \log P(X|z) - \log P(z)]$$

Rearranging the sign, we get our objective function:

$$\begin{aligned} \log P(X) - D_{KL}[Q(z|X)\|P(z|X)] &= E[\log P(X|z) - (\log Q(z|X) - \log P(z))] \\ &= E[\log P(X|z)] - E[\log Q(z|X) - \log P(z)] \\ &= E[\log P(X|z)] - D_{KL}[Q(z|X)\|P(z)] \end{aligned}$$

Hence $Q(z|X)$ is the encoder and $P(X|z)$ is the decoder. Variational autoencoder combined with Long Short memory network is the baseline in this paper to obtain the lowest metrics defined later below.

6 Prediction using a Long short-term neural network

This paper aim to provide reliable predictions of expected returns $E(X_{t+1})$ to explain most of the forecasted variance among assets of the universe. To predict $E(X_{t+1})$, we use hypertuned Long Short term memory (LSTM) neural network (Hochreiter, S. and Schmidhuber, J.,1997) on the next interval of time. LSTM is designed to overcome recurrent neural network in a way that it remembers past information and its conditions based on past values. LSTM are also able to handle noisy data as is financial markets which it makes suitable for our analysis.
Given recent research (Sirignano J. and Cont R., 2018), an optimal sampling size or path-dependence is denoted with as much data as possible to improve out-sample forecasting accuracy. Nevertheless, results remain robust and stationary over time according to the authors.
Strictly speaking, the mathematics behind the LSTM cell (the gates) is:

$$g_t = \tanh(X_t W_{xg} + h_{t-1} W_{hg} + b_g),$$

$$i_t = \sigma(X_t W_{xi} + h_{t-1} W_{hi} + b_i),$$

$$f_t = \sigma(X_t W_{xf} + h_{t-1} W_{hf} + b_f),$$

$$o_t = \sigma(X_t W_{xo} + h_{t-1} W_{ho} + b_o),$$

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t,$$

$$h_t = o_t \odot \tanh(c_t),$$

where $\odot$ is an element-wise multiplication operator, and, for all $x = [x_1, x_2, \ldots, x_k]^\top \in R^k$ the two activation functions:,

$$\sigma(x) = \left[\frac{1}{1+\exp(-x_1)}, \ldots, \frac{1}{1+\exp(-x_k)}\right]^\top,$$

$$\tanh(x) = \left[\frac{1-\exp(-2x_1)}{1+\exp(-2x_1)}, \ldots, \frac{1-\exp(-2x_k)}{1+\exp(-2x_k)}\right]^\top$$

6.1 Our approach

In this paper, we divide each dataset into training set, validation set, and testing set at 50/20/30% of the total number of rows after feature creation. We follow by training VAE with Long Short-term deep neural network on the training dataset with hyperparameter tuning on the validation dataset through Bayesian Optimization. We continue to obtain hyper tuned predictions for the testing dataset and evaluate the performance of each of these predictions.

6.2 Results

We finally compute the overall performance of the crypto-currency strategy through directional accuracy criteria. We obtain 50.62% on order book data to predict the next minute. However on the highest quantiles of predictions, an accuracy of 53% is achieved.

7 Conclusions

- VAE and LSTM resulted in good prediction accuracy.
- The rough path signatures managed to capture some of the structure in the data and gave decent predictions when using logistic regression.
- Prediction using dilated CNN gave an accuracy similar to random guessing.
- The GAN features could not explain the data when used together with dilated CNN for prediction.

8 Project files

All the files needed to run the analysis of this report are available in the Google Drive folder here.

References

[1] Time series forecasting with convolutional neural networks - a look at wavenet. https://jeddy92.github.io/JEddy92.github.io/ts_seq2seq_conv/m. Accessed: 06/10/2019.

[2] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 214–223, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

[3] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014.

[4] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5767–5777. Curran Associates, Inc., 2017.

[5] Lajos Gyurkó, Terry Lyons, Mark Kontkowski, and Jonathan Field. Extracting information from the signature of a financial data stream. 07 2013.

[6] B. Hambly and Terry Lyons. Uniqueness for the signature of a path of bounded variation and the reduced path group. *Annals of Mathematics*, 171, 07 2005.

[7] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W. Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *CoRR*, abs/1609.03499, 2016.

[8] Weixin Yang, Lianwen Jin, Hao Ni, and Terry Lyons. Rotation-free online handwritten character recognition using dyadic path signature features, hanging normalization, and deep neural network. pages 4083–4088, 12 2016.