Predicting S&P 500 Returns Using Gramian Angular Field and Multiple Input CNN's

Team Jonas contribution to ERP Prediction Contest, February 15, 2019 - May 15, 2019

Hull Tactical, University of California Santa Barbara Department of Statistics and Applied Probability and the Center for Financial Mathematics and Actuarial Research

Jonas Lundgren

University of California Santa Barbara

jonaslundgren94@gmail.com

May 7, 2019

ABSTRACT

Image representations of time series data are created using Gramian Angular Field. The image representations are fed into a multiple input Convolutional Neural Network which are used to make S&P 500 returns predictions. Different features and number of features were tested to make predictions on out of sample data. The model resulting the largest R^2-score was a single input network with the image representation of the percentage change from day to day of S&P 500 closing prices as input.

1 Introduction

This is Jonas contribution to 2019 ERP contest. The goal of the competition is to predict the 5 (trading) day forward ERP of the S&P 500 adjusted for dividends (ASPFWR5) based on historical data. More details on the competition can be found on the competition website here: `http://erpcontest.pstat.ucsb.edu`. Code can be found at: `https://github.com/LurreMcFly/ERP-PREDICTION-CONTEST/`.

The idea of this contribution is to transform time series data into images and use a regression Convolutional Neural Network (CNN) to make predictions of ASPFWR5. Usually other network types such as recurrent neural networks and especially Long Short Term Memory Networks are used for time series prediction. Since CNN's have a great track record in problems related to computer vision, it would be interesting to try to use them for time series prediction.

A problem that arises when using a CNN for time series prediction is to create meaningful image representations of the time series. One way of doing it, is with a method called Gramian Angular Field, described in the paper *Imaging Time-Series to Improve Classification and Imputation* by Wang and Oates [1]. This method has, in other areas, shown to produce great results.

When using Gramian Angular Field as image representation, each image represents a single time series i.e. the time series for one variable, for some time steps N back in time. Since we want to use several variables in order to make predictions we are going to feed the CNN more than 1 image at a time. For the prediction part of the competition, Jonas contribution fed 1 image at a time to the network, images of the closing prices, to predict the ASPFWR5. Here models taking several variables (images) into consideration when making the prediction will be investigated. The models presented in this project uses one, almost complete, CNN per image and merges the output from each CNN into one last neural network that makes the final prediction. To load data, train and build the network *Python*, *Fastai* and *Pytorch* were used and computations were run on a cloud GPU through: `https://crestle.ai/`.

Different features and number of features were used to make a prediction are tested on out of sample data. The model resulting the largest R^2-score is the single input network with the image representation of the percentage change from day to day of S&P 500 closing prices as input.

1.1 Data

The data set consists of observations of the S&P 500 price and a large number of predictors from 1952 to 2017, it can be found at the competition site [2]. Every 11th year is set aside for validation, to keep us from overfitting the model. The choice of setting aside each 11th year is made arbitrarily with only reason that 11 being a prime number and thus hoping to prevent any repetition in time.

The year 2017 is also set aside for final testing of the models. Training of the models are done using the remaining data.

2 Procedure

The prediction of ASPFWR5 could be divided up into three parts. The first part being what features to use in the model, section 2.1 Feature Engineering. The second part consists of creating the image representation of the time series, section 2.2 Gramian Angular Field. The third part the building, training and usage of a CNN to make the predictions, section 2.3 Convolutional Neural Network. The total procedure is described in Figure 1.

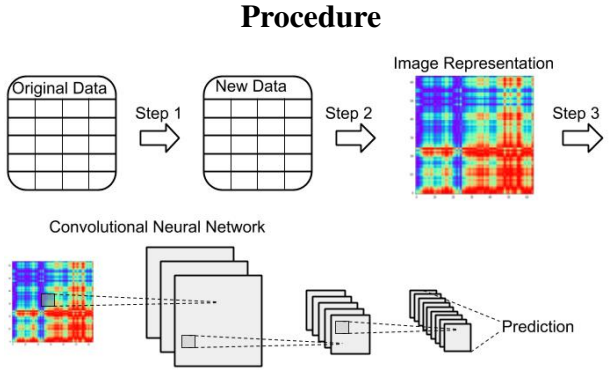

Figure 1: The process, from data to prediction.

2.1 Feature Engineering

Features were selected based on a previous project in swing trading, due to the 5 day prediction period. The added features were based on: 1. *Relative Strength Index* (RSI), 2. *Moving Average Convergence/Divergence* (MACD), 3. *Bollinger Bands* and 4. *50 and 200 Days Moving Average*. All which are measures of the either: the price, the volatility of price or both. For each of the features there are parameters that could be tinkered with, however no testing of different parameters when constructing the above mentioned features was been done.

It should be mentioned that the the feature based on *Bollinger Bands* are not the actual Bollinger Bands given by an interval of a upper $\texttt{BB}_{upper}$ and lower $\texttt{BB}_{lower}$ band. Where $\texttt{BB}_{upper} = \text{MA}(30) + 2\text{sd}(30)$ and $\texttt{BB}_{lower} = \text{MA}(30) - 2\text{sd}(30)$, where $\text{MA}(30)$ is the 30 day moving average for the last 30 days and $\text{sd}(30)$ the standard deviation of the last 30 days[5]. Instead a normalized difference between the mean of the bands was used i.e.

$$\texttt{BB}_{feature} = \frac{\texttt{CLOSE} - (\texttt{BB}_{upper} - \texttt{BB}_{lower})/2}{(\texttt{BB}_{upper} - \texttt{BB}_{lower})}.$$

Here $\texttt{BB}_{feature}$ is the feature we are going to use in the model and CLOSE the closing price for corresponding day.

It should also be mentioned that the feature based on moving average is the difference between the 50 and 200 days moving average.

2.2 Gramian Angular Field

We want to represent the different time series of our predictors as a two dimensional image. We are going to make use of the angle relationship between the different time points. The time series is first scaled with min-max scaling to the interval $[-1, 1]$. Each observation in time has two variables: time of observation and corresponding value for time series. The angle and radius are thus expressed in the following way:

$$\begin{cases} \phi_i = \arccos(x_i) \\ r_i = \frac{i}{N} \end{cases}$$

where x_i is the value for the time series at observation $i = 1, ..., N$ where N is the total number of time steps used. We have thus expressed the time series in polar coordinates with as a series of radiuses r_i and angles ϕ_i. The choice of N is important as it is how many time steps back in time we are going to take into consideration when making the prediction. For the models used in this report $N = 64$. The choice is loosely based on the presentation by Blair Hull, which can be seen on the contest web page [2], where three months look back was mentioned. This in combination with $64 = 2^6$ being a power of 2 which being suitable as input for the CNN, are the reasons that $N = 64$ was chosen.

The next step is to calculate a matrix with sinus of the difference between the angles in the polar coordinate representation i.e. $\sin(\phi_i - \phi_j)$ for $i, j = 1, ..., N$. When $i = j$, the diagonal of the matrix is considered a special case and it contains the original value/angle for the time series. The matrix of sinuses is the image representation. Figure 2, taken from Wang and Oates [1], displays the process of creating the angular field.

Creating a Gramian Angular Field

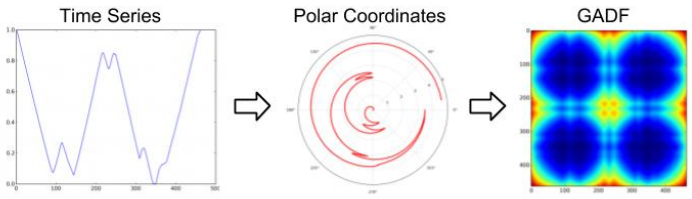

Figure 2: The process of creating the Gramian Angular Field representation of a time series.

The image representation of the different features described previously, can be seen in Figure 3. The slow moving *MA* based on the difference between the 50 days moving average and the 200 days moving average looks smooth especially compared to the image representation of the fast moving *percentage*.

Different Features on 1/3/2017 Represented Using Gramian Angular Field

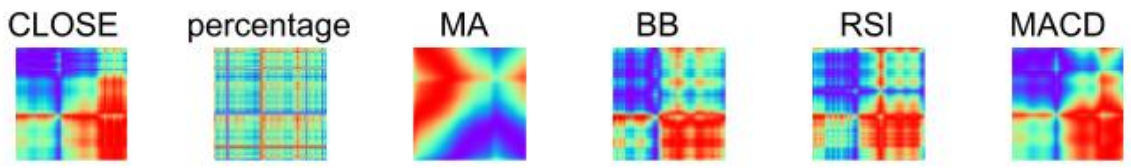

Figure 3: Gramian Angular Field Representation of the Different Features on 1/3/2017. CLOSE is the closing price, percentage the percentage shift from previous day, MA is the difference between the 50 days moving average and the 200 days moving average. BB the $\texttt{BB}_{feature}$ based on Bollinger Bands. RSI the Relative Strength Index and MACD the Moving Average Convergence/Divergence.

How the Gramian Angular Field representation shifts over time is represented in Figure 4.

Gramian Angular Field Representation Over Time

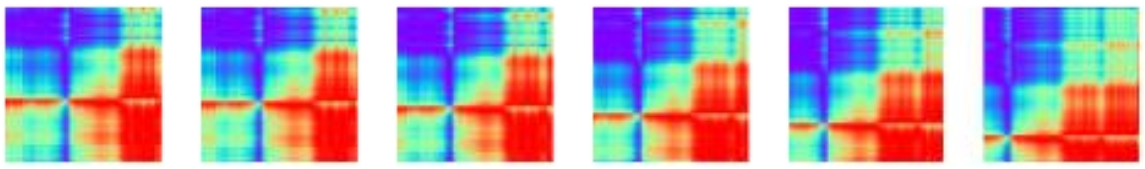

Figure 4: Sample of Gramian Angular Field Representation of the feature `CLOSE` over 16 trading days from 1/3/2017 to 1/25/2017.

2.3 Convolutional Neural Network

For the case of two inputs, the neural network consists of tree separate networks, and can be seen in Figure 5. The two inputs are noted as *Predictor1* and *Predictor2* in Figure 5, are the *Gramian angular field* representations of, the daily closing price and daily percentage change, respectively for a window of 64 days.

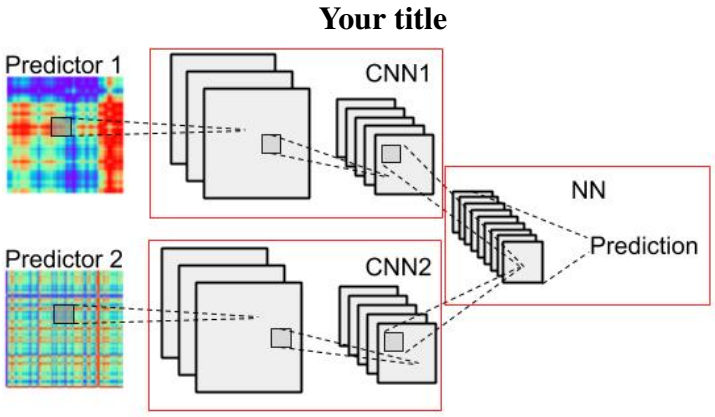

Figure 5: CNN architecture for predictive model with 2 predictors.

Each of the predictors are fed into a separate CNN noted as CNN1 and CNN2 which both are of the network structure *ResNet50* [3], where the last layer of the network has been removed. The last layers from the two networks are combined to be able to take input from both CNN's and it make the final predictions, it is noted as NN if Figure 5 .

For each feature a single input network is previously trained and saved. When assembling the three networks in Figure 5, the pre-trained weights for the single input networks are loaded into the double input network. That is, in Figure 5, CNN1 and CNN2 have both been separately trained previously for *predictor 1* and *predictor 2*. When assembling the whole network the weights are first loaded into CNN1 and CNN2 before cutting of the last layers from both networks. The weights from the last layer for both CNN1 and CNN2 are stored and loaded in to NN which is added lastly. NN therefore consists of one linear layer loaded with the weights of the cut off last layers from previous two CNN's. If for example the two networks CNN1 and CNN2, before the cut off, each had a last layer with 32 weights the last NN will have 64 weights where the first 32 weights are from CNN1 and the last 32 weights are from CNN2. The last layer has one output which is the prediction.

The same procedure is used for more inputs and for single inputs the network structure *ResNet50* was used.

3 Results

The results presented in Table 1 are based on the test data from year 2017. The metric used to evaluate model in the competition was $R^2 = 1 - \frac{\sum_i (y_i - f_i)^2}{\sum_i (y_i - \bar{y}_i)^2}$. In Table 1 the mean squared error (MSE) as well as the R^2 are shown for each model. The model resulting the largest R^2 for the out of sample data is the CNN with the single input time series of the percentage change from day to day.

Table 1: Table of Results

Features included	**Training MSE**	**Training R^2**
CLOSE	6.1964e-4	-0.2549
Percentage	**5.8410e-4**	**-0.1347**
CLOSE & MA20050diff	7.3129e-4	-0.5069
Percentage & RSI	6.2992e-4	-0.2772
Percentage & BB	5.9430e-4	-0.1413
CLOSE & Percentage & MA20050diff & RSI	6.8879e-4	-0.4272
Percentage & BB & MA20050diff & MACD	7.4752e-4	-0.6261

Table 1: The results of predictions on ASPFWR5 using the CNN models with the features described in the column Features based on the test set from year 2017. The results are given as the mean squared error (MSE) and the R^2. The model resulting the largest R^2 for the out of sample data is the CNN with the single input time series of the percentage change marked in bold font.

4 Discussion

I thought the result of the models would improve as more features were taken into consideration. I think the explanation of why it did not, has to do with one or more of the things I would like to address in regards to the models. These are the features, the image representation, the CNN structure and the training process.

It is worth mentioning that process of having to create image representations and training networks made it hard to test different parameters for the features and more features in generall.

4.1 Features

Most of the features provided with the data set [2] were left out of the model and instead technical indicators were used. From a prediction point of view I think this was a mistake when trying to make the best possible prediction. More testing with the features provided could definitely be done.

4.2 The Image Representation

The problem of creating relevant image representations still remains. I think the Gramian Angular Field approach might introduce unwanted noise in the image. Since CNN's are good at detecting shapes I would suspect that using some other image representation with more focus on creating edges and contours would be a good place to start. This might be a reason why the feature percentage, which resulted in images with more sharp edges, gave the best results.

It would have been favourable to include several features in the same image representation. This would result in a model where we did not having to stitch several CNN's together and problems associated with that could be avoided.

4.3 CNN Structure

If we could represent different time series in one image representation we would avoid the problem of how to combine multiple CNN's together. Here only the last linear layer was replaced for each network with one larger linear layer. More experimentation could be done using more layers or merging the different CNN's at an earlier stage than in the last layer.

Another approach than what was used in this project would be to stack several image representations on one another. The input would instead of an (64,64,3) for the tensor be a (64,64,6) tensor if two images were stacked on top of each other. Other problems would then have to be dealt with, such as constructing the kernel of the first layer. It would be interesting to see the results of that approach.

4.4 CNN Training

Due to the special network structure, the training of the network became harder. There are built in functions for finding appropriate learning rates when training the network. These functions did not work as intended when having multiple inputs, due to each of the CNN's having their own optimal learning rate. The normal pattern, of an decreasing training and validation error until a point where the validation starting to increase, was not observed for networks with multiple inputs. Instead the different errors would jump all over the place. I therefore suspect not all networks reached their optimal potential.

4.5 Closing Thoughts

All together, my conclusion is that there are more optimal ways/models of predicting the S&P 500 Returns. However I think there are fields (maybe in finance as well) where this modelling approach could be used. The creation of image representations of non-image data gives another

way of understanding of the data. The multiple input network could be used in other ways as well when different predictors of different types are working together to make a single prediction.

I would lastly like to thank both Hull Tactical and UCSB department of statistics and applied probability for hosting this competition. The competition gave me a reason to learn things I wanted to learn, and I have developed new skills thanks to you. Thank you!

References

[1] Zhiguang Wang and Tim Oates. Imaging Time-Series to Improve Classification and Imputation. *arXiv:1506.00327v1*, 2015.

[2] `https://ucsb-erp-contest.herokuapp.com/`

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun and Microsoft Research Deep Residual Learning for Image Recognition *arXiv:1512.03385v1*, 2015.

[4] https://en.wikipedia.org/wiki/Bollinger_Bands, Bollinger Bands, 2019

[5] `https://github.com/LurreMcFly/ERP-PREDICTION-CONTEST/`,