FINANCIAL TIME SERIES FORECASTING

Group 15, April 2021

Abstract

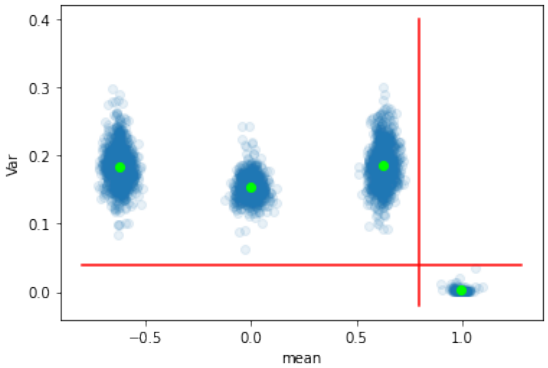
The task is to predict the stock returns on the 51^{st} day given a 50-day time series. Since there is only one variable to be predicted, returns, this work will focus on two state-of-the-art techniques for Univariate Time Series Prediction Problems: Support Vector Regression and Recurrent Neural Nets. For clarity, this report is divided into the following sections: Data Exploration, Model Selection, Data Pre-Processing, Hyperparameter Tuning, and Results.

1 Introduction: Overview of our Approach

There is a noticeable substantial difference between two classes of assets which can be seen by looking at Asset returns over the time dimension. These different classes will be further denoted as Optimal and Volatile. Subsequently, two separate models can be developed respectively for predicting returns taking into account this difference.

2 Data Exploration

By plotting the data, it can be seen that there is a set of Assets with higher mean (returns) and lower variance (volatility).



There were only a few samples with higher mean and lower variance. This makes sense, as in Finance this translates into higher returns with lower volatility, which is quite rare to find. Therefore, the scarcity of such data points (4% of the whole dataset) is justified and the hypothesis that they are outliers can be disregarded.

2.1 Clustering

A Gaussian Mixture Model would produce good results for $k = 4$ clusters. However, for $k = 2$ the decision boundary is quite evident from the mean-variance plot in this case: all time series with mean centered at 1 had $\mu > 0.8$ and $\sigma^2 < 0.04$. As a result, an Indicator function was built to discriminate which points belonged to each cluster.

3 Model Selection

3.1 Model for Optimal Assets

Due to the very low variance in this dataset using the mean as prediction for all instances could have been enough to get a close-to-optimal precision. But, a slightly better precision is achievable by applying the SVR model

which is more robust to outliers[1], thanks to its ϵ-tube: points within a distance ϵ from the hyperplane are not penalized in the training.

3.2 Model for Volatile Assets

The larger sample (order of 10^4), makes it possible to train a Recurrent Neural Network on the Volatile class. Therefore, this more sophisticated model was applied with the purpose of recognizing the patterns in more volatile data.

4 Data Pre-Processing

4.1 Support Vector Regression

This model does not constrain too much the shape and the distribution of the input and delivers a good result even if fed with raw data, due to its very good generalization capability.

4.2 Recurrent Neural Network

Usually with LSTM Recurrent Neural Networks, if quantities are spread across a wide range of values, it makes sense to rescale them [2]. However in this case the range is just (-1.93,1.94), very close to scaled in the interval (-1,1). This explains why scaling the data does not improve the performance.

5 Hyperparameter Tuning

5.1 SVR

Running a Random Search we restrict the parameter space to a proper subset of it where the cross validation error is minimized, so that a GridSearch can be run in a reasonable time inside this subset to find even 'more optimal' parameter values, which are then selected for the final training.

5.2 RNN - Stacked LSTM

The architecture taken was based on the notes from A.Géron's book[3], which is:

- one input unit
- two LSTM hidden units, with an unknown number of neurons X_1, X_2
- one Dense output unit, with a single neuron since the output $r \in \mathbb{R}$ is univariate

Therefore, the number of neurons for each hidden layer and the learning rate are the parameters to optimize by using the Hyperband tuner from the Keras library. After fitting the model with the optimal number of neurons, it was noticed that the model was overfitting. So, to prevent this from happening, Dropout Layers were added. To improve the performance further, K-fold Cross validation could have been firther implemented.

6 Results

Finally, the models were trained and the solutions merged:

6.1 Optimal Assets MSE

Mean Square Error of our fitted SVR model converges to 0.0001106 on the test set.

6.2 Volatile Assets MSE

Mean Square Error of our fitted stacked LSTM model converges to 0.015463 on the test set.

6.3 Overall MSE

Mean Square Error of our merged model converges to 0.01586 on the submission test set.

7 Bibliography

References

[1] Raj, A. *Unlocking the true power of support vector regression.* from https://towardsdatascience.com/unlocking-the-true-power-of-support-vector-regression-847fd123a4a0 ,(2020, October 05).

[2] Brownlee, J. *How to scale data for long short-term memory networks in python.*. Retrieved April 27, 2021, from https://machinelearningmastery.com/how-to-scale-data-for-long-short-term-memory-networks-in-python/

[3] Géron, A. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems (2nd ed.).* O'Reilly (2019).

[4] Sethi, A. *Support vector regression in machine learning.* from https://www.analyticsvidhya.com/blog/2020/03/support-vector-regression-tutorial-for-machine-learning/ ,(2020, April 01).