FINANCIAL TIME SERIES FORECASTING

Group 15, April 2021

Abstract

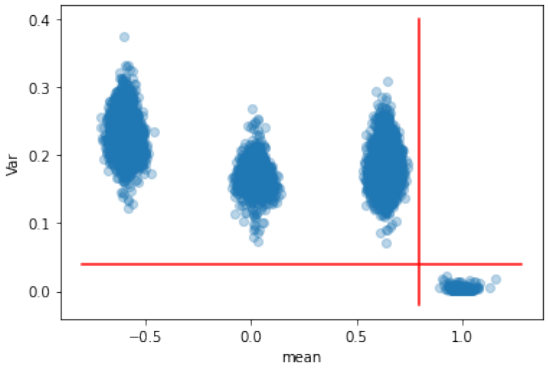
The task is to predict the stock returns on the 51^{st} day given a 50-day time series. Since we only have to predict one variable, returns, we focus on two state-of-the-art techniques for Univariate Time Series Prediction Problems: Support Vector Regression and Recurrent Neural Nets. For clarity, we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

1 Introduction: Overview of our Approach

Looking at Asset returns over the time dimension we noticed a substantial difference between two classes of Assets, which we will denote as Optimal and Volatile. Subsequently, we trained a different model on each class for predicting the target variable.

2 Data Exploration

By plotting our data, we can see that there is a cluster of Assets with higher mean (returns) and lower variance (volatility).



We could have used a K-means Clustering or Gaussian Mixture Model to group the data points into classes. However, Since the decision boundary is evident from the plot, we used the fact that time series with mean centered at 1 had all $\mu > 0.8$ and $\sigma^2 < 0.04$. As a result, we built an Indicator function to discriminate which points belonged to each cluster. 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 we can disregard the hypothesis that they are outliers.

3 Model Selection

3.1 Model for Optimal Assets

Due very low variance in this dataset we could get a close-to-optimal precision even by just using the mean as prediction for all the instances, but a Support Vector Regression (SVR) model will give a slightly better precision and will also be more robust to outliers [1]

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. Hopefully this more sophisticated model will be able to recognize 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 good result even if we just put the raw data inside, 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. 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 we can run a GridSearch in a reasonable time inside this subset to find even 'more optimal' parameter values, which we select for the final training.

5.2 RNN - Stacked LSTM

For the architecture we took inspiration from A.Géron [2]

- 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

The parameter that we try to optimize then is the number of neurons for each hidden layer, and the learning rate. In order to do that we use the Hyperband tuner from the Keras Library. After fitting the model with the optimal number of neurons, noticing that the model was overfitting, we add Dropout Layers to prevent this from happening. To improve our performance further, we should have used K-fold Cross validation

6 Results

Finally we train the models and merge the solutions:

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 0.01583 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

[1] Brownlee, J. (2019, August 05). 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/

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

[3] Sethi, A. (2020, April 01). Support vector regression in machine learning. Retrieved April 27, 2021, from https://www.analyticsvidhya.com/blog/2020/03/support-vector-regression-tutorial-for-machine-learning/

[4] Raj, A. (2020, October 05). Unlocking the true power of support vector regression. Retrieved April 27, 2021, from https://towardsdatascience.com/unlocking-the-true-power-of-support-vector-regression-847fd123a4a0