Visually-Good and Useful Stock Prices Forecasting

September 8, 2021

Mirko Giacchini

Abstract

We considered the problem of predicting the next stock price of an asset from the sequence of previous prices. We observed that using the actual stock prices as data prior we are able to predict the general trend of the prices but the predictions are not so good when used as part of a trading strategy, viceversa using the differences of adjacent prices gives good models for the trading strategy but with bad predictions for the actual prices. We propose a loss function that attempts to take the best out of these two strategies. The source code is available on github.

1. Introduction

The stock prices of an asset are extremely volatile over time and thus represent a good opportunity to asses the performances of forecasting models. It has been shown that deep learning seq2seq models outperforms classical methods used in finance (Mehtab et al., 2020; Zou & Qu, 2020). In particular in (Zou & Qu, 2020) it has been shown that deep learning models performs better than classical methods with respect to standard regression metrics like MSE (meaning that the DL models predict better the actual prices) but also when the model's predictions are used inside a trading strategy (meaning that the predictions from the DL models are also useful in practice).
Our goal is to build models that maximize the money gained by the trading strategy while giving a good visual prediction of the real prices. We'll consider the following trading strategy based only on the asset's open price: at the beginning of each day if the predicted open price for the next day is greater than the open price of the current day, we buy at the open price of the current day and sell at the open price of the next day (note that this is equivalent to buying at the predicted local minimums and selling at the predicted local maximums). To achieve our goal we'll use a loss different from the classical MSE, note that there are already several works which propose new loss functions to improve the performances of seq2seq models (Le Guen & Thome, 2019; Lee, 2007).

Email: Mirko Giacchini <giacchini.1811809@studenti.uniroma1.it>.

Deep Learning and Applied AI 2021, Sapienza University of Rome, 2nd semester a.y. 2020/2021.

2. Datasets

We will use three datasets from yahoo finance corresponding to three assets: Google (around 17 years of daily prices), Apple (around 40 years) and Bitcoin (around 7 years). We always use the last year as test set and the second last year as validation set, the rest is used for training. We'll use only the open price itself as feature. We'll use min-max normalization of the values as preprocessing step.

3. Models

We'll tackle the problem as an autoregressive task: given a sequence of prices we'll predict the sequence shifted by one (so we also introduce the price for the next day). RNNs and LSTMs in particular have proved themselves to be really powerful on autoregressive tasks, also in the financial context. We will therefore use a standard LSTM model with one layer and followed by a shared linear layer to make the final prediction.
Recently it has been shown that attention is enough to build strong seq2seq models (Vaswani et al., 2017), therefore we will also use a model inspired by the classical transformer architecture. In the classical architecture the input for the transformer encoder is an embedding of the elements summed with a positional encoding, however we'll use a LSTM to learn the embedding of the elements (which already considers the positions); moreover we'll replace the transformer decoder with a simple linear layer (shared across all elements of the sequence), the encoder is instead the one described in the original Transformer paper.

4. Loss Functions

Let $X = x_1, x_2, \ldots, x_n$ be a sequence of open prices. We empirically found out (experimenting on a dataset not included in the ones mentioned before) that using the classical MSE loss and the normalized sequences of prices (ie: sequences like X, normalized) as data prior we obtain visu-

ally convincing predictions of the actual prices, but modest gains from the trading strategy. On the other hand if we use sequences of differences during training (ie: we use $x_2 - x_1, x_3 - x_2, \ldots, x_n - x_{n-1}$ instead of X, again normalizing the elements) and MSE loss then we obtain an higher gain from the trading strategy, but the actual prices reconstructed from the predictions (note that now the predictions are of the differences) are extremely different from the real values.

We propose a loss to be used with sequences of open prices (ie: X normalized) that attempts to improve the gain of the models while mantaining visually good predictions.

To ease the notation we're writing the losses for only one sequence (of length n), the generalization to a batch of sequences is straightforward. The general form of our loss is $\ell_\theta(y, \hat{y}) = mse(y, \hat{y}) + \lambda \cdot d(y, \hat{y})$ where $mse(y, \hat{y}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$ is the classical MSE loss, $d(y, \hat{y})$ penalizes errors in the differences and λ is an hyperparameter which controls how much importance the differences have with respect to the real values. A natural choice for d is $d_{v0}(y, \hat{y}) = \frac{1}{n-1}\sum_{i=1}^{n-1}(z_i - \hat{z}_i)^2$ where $z_i = y_{i+1} - y_i$ and $\hat{z}_i = \hat{y}_{i+1} - \hat{y}_i$. Intuitively this loss should take advantage of what we observed in pratice: training to minimize the MSE on the differences helps in increasing the gain from the trading strategy. However, experimentally, the behaviour of a model trained only with d_{v0} loss is very different from one trained on the differences used as data prior: the reason is that in the second case the differences are normalized, instead in the first case we are normalizing the prices and this doens't directly normalize the differences. The intuitive idea to fix the issue is to include a correction inside the loss function to deal with the normalization. A first idea could be to invert the normalization of the y's and normalize the z's directly in the loss function, however this would make the loss dependent on the normalization chosen, instead we would like a more general approach.

We conjecture that normalizing the prices has the effect of making the model less sensitive to sudden changes, thus helping in predicting the actual prices but making the training harder when we want to minimize the error on the differences. Based on this conjecture we propose to increase the values of the differences inside the loss function in order to make them more pronounced, in particular instead of considering $z_i - \hat{z}_i$ we consider $e^{z_i} - e^{\hat{z}_i}$ inside the loss. Note that $z < e^z$ for all z: the differences are being increased. To avoid too large values in the loss we apply a $tanh$ in order to have values in $[-1, 1]$: we'll have $tanh(e^{z_i} - e^{\hat{z}_i})$ instead of $z_i - \hat{z}_i$. The final d loss is then: $d(y, \hat{y}) = \frac{1}{n-1}\sum_{i=1}^{n-1}(tanh(e^{z_i} - e^{\hat{z}_i}))^2$. Note that d is minimized when the argument of the $tanh$ is 0, so we are forcing $z_i \approx \hat{z}_i$ for all i.

5. Results and conclusion

We trained our two models on the three datasets using all the three possible losses. To quantify the quality of the predicted curves we use the r2-score and to measure the quality of the models when used for the trading strategy we consider the ratio between the gain of the model and the optimal gain. The results are summarized in the tables below ("MSE diff" is the model trained on the normalized differences, MSE+D is our loss). Note that, as expected, the gain of "MSE diff" is always larger than the one of MSE, and moreover our loss always improve the gain of MSE (in the Google's dataset the gain is even larger than "MSE diff"). Recall that our second goal is to preserve the curves of standard MSE, unfortunately the behaviour seems less stable in this case, note for example that on the Apple's dataset the transformer with MSE+D loss has an enormous drop in the r2-score. However in all other cases the r2-score is reasonable (in some cases it even increases), so we can conclude that in most cases our loss acts as expected. In figure 1 we show a significative example of the effect of our loss.

Table 1. Results on Bitcoin's dataset

		MSE	MSE diff	MSE+D
LSTM	gain	3.27%	10.35%	10.21%
	r2	76.6%	-85.18%	71.23%
Transformer	gain	6.14%	12.8%	11.88%
	r2	56.24%	-602.8%	21.83%

Table 2. Results on Apple's dataset

		MSE	MSE diff	MSE+D
LSTM	gain	4.98%	10.7%	10.37%
	r2	73.6%	72.66%	90.82%
Transformer	gain	8.38%	20.51%	10.25%
	r2	95.36%	81.11%	-2.36%

Table 3. Results on Google's dataset

		MSE	MSE diff	MSE+D
LSTM	gain	9.2%	14.42%	14.67%
	r2	88.59%	-311.8%	85.89%
Transformer	gain	5.43%	15.94%	16.07%
	r2	98.46%	-53.23%	99.33%

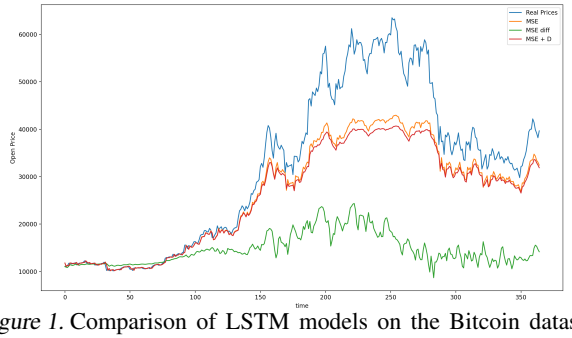

Figure 1. Comparison of LSTM models on the Bitcoin dataset, note that MSE+D has almost 7% more gain than MSE

References

Le Guen, V. and Thome, N. Shape and time distortion loss for training deep time series forecasting models. 2019.

Lee, T.-H. Loss functions in time series forecasting. 2007.

Mehtab, S., Sen, J., and Dutta, A. Stock price prediction using machine learning and lstm-based deep learning models. 2020.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pp. 6000–6010, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

Zou, Z. and Qu, Z. Using lstm in stock prediction and quantitative trading. 2020.