ML Nanodegree Capstone Project – Proposal

Maciej Nalepa

2020 January

1 Background – Investment and Trading

Knowing the future of a market stock price is a very valuable information about risk of investment and is commonly pursued because of potentially infinite profits.

Unfortunately market is affected by very large amount of factors that are difficult to implement all together. We can try using NLP on news to gather possible changes on the market or try a technical analysis and many more. Learning how to efficiently forecast a price requires finding patterns, especially "anomalies" that may be used to get a probable future value (James Simons, Numberphile 2015). ML is a great tool for finding patterns in big data and we can use it to solve this forecasting problem.

Currency ratio and stock markets forecasting based on technical analysis can be easily implemented in a script thanks to timeseries prediction nature of the problem. Price and volume are publicly accessible and can be used to generate a set of rules which are aiming to output the most likely future price.

My personal experience was to work on trading bots myself for a few months. I had learned that trading bot efficiency is short lived and extremely prone to overfitting to historical data which makes them useless in real world predictions applications. However, all of my work was developed using hard coded rules or simple linear models. I want to use gained knowledge to improve with a far more advanced model.

2 Problem statement

We want to predict the price of EURUSD ratio and NASDAQ stock. This is a time-series forecasting problem. The goal is to create a script that will forecast future values basing on input provided as starting point. Then a web application will serve forecasts live by downloading information from third party services.

3 Dataset and inputs

Our data will consist of price candles entries with columns:

- datetime – date and time of the price candle opening (may be split into separate columns),
- open – price at the candle entry time,
- high – maximum price reached during candle lifespan,

- low – minimum reached during candle lifespan,
- close – candle exit price.

This data can be used to calculate indicators for example: Momentum, MA, OsMA, MACD. Indicators can be used as additional features.

Candle lifespan will range from 1 minute to 1 day (if provided only in shorter lifespan a longer will be calculated).

Data is stored in `.csv` format and to keep it small size I will include the history back to year 2015. Time span of 5 years should be more than enough to train a model.

Dataset will be split into 80% train, 20% validation and test 20%. Of course data has to be stored chronologically so subsets will be exactly in that order.

Data sources

- `https://finance.yahoo.com` – live and historical data
- `http://histdata.com` – historical data

4 Solution statement

To solve this problem I will use PyTorch. We will build an LSTM model and an fully connected model with a 1D Convolution layer as an extension to our features: preprocessed stock value, selected indicators.

5 Benchmark model

Moving Average as proposed here will be used as reference model.

6 Evaluation metrics

MSE between predicted y and actual values $\hat{y}$:

$$MSE = \frac{1}{2}(y - \hat{y})^2 \tag{1}$$

Model loss will be calculated from validation set that will be less or equal 20% of the actual values that a model tried to predict from a starting point. This means that up to 40% tail of the dataset is reserved only for evaluation during training and will not be fed as input. For final model testing the actual values from validation set will be included as input to predict test actual values.

7 Project design

7.1 Normalization

Crucial part of the design is to normalize the data. Stock values can range from 0 to infinity, but we cannot tell our model explicitly about it and data suggests that upper

limit is the maximum value from the dataset. There are at least 3 ways of solving this problem:

- 1-D Convolution layer – 1×3 shape trainable filters, reverse cannot be easily implemented, an alternative solution is to apply a linear transformation with trainable coefficients,
- 1-D Batch Normalization layer – trainable parameters Δ and σ: $norm = \frac{mean+\Delta}{\sqrt{variance+\sigma}}$, reverse of this layer is easy to implement,
- MA indicator – $norm = \frac{price-\mathrm{MA}}{\max(price)}$ where $\mathrm{MA} = \frac{\sum_{i=0}^{\mathrm{length}} price_{\mathrm{close};i}}{\mathrm{length}}$ requires length parameter that will become a hyperparameter of our model, reverse of this method is straightforward (notice that $i = 0$ is the most recent value).

Important: Because this is a regression model, every normalization should be reversed if possible. Otherwise the model has to solve reversing of normalized data itself, which can reduce accuracy drastically.

Note: Some indicators are normalized from their design. These are called oscillators and include e.g. RSI, Stochastic.

7.2 Architecture

Input can consist of some recent price values – minimum 3 candles. To keep the model simple we can predict only the close price.

Best model architecture has to be experimentally discovered. I will start with single unit LSTM to predict just next price value. It may prove that more units allow better prediction once the forecast length increases (forecasting more values during single run).

7.3 Script

Will run prediction on any input provided (in correct shape obviously) with steps forward specified as extra parameter (will automatically rerun model with input extended by a predicted value until expected steps are reached).

7.4 Web-app

Site will run prediction every time it gets visited, loading cached output if there is no update necessary. Input will come from Yahoo! Finance. Amazon SageMaker Endpoint will be used to deploy the model.