Stock Market Trading with LSTM Analysis

Franklin Doane
dept. of Computer Science
Tennessee Technological University
Cookeville, United States
fgdoane42@tntech.edu

I. Dataset Creation

A. Find stocks by market cap

I downloaded a list of all stocks ranked by market cap off the internet

B. Select largest stocks with sufficient data

Next, I used the list along with the yahoo finance api to select the 1500 most valuable companies that satisfied the following conditions: data for training dating back to the beginning of 2011, and additional data before that sufficient for indicator calculation and for creating the first time-series input with Jan 01 2011 as it's last date.

C. Download train and test data

I used 10 training datasets, the first one containing data for all market days in 2011-2012, with each following adding 1 sequential year of data. There are 10 evaluation sets, each contain data for a single year, from 2014 - 2023. Each training set and testing set downloaded data for every stock in the list using the yahoo finance api. All data is made up of 1 day candlesticks. Each download with the size of the dataset, with additional older data sufficient for calculating indicators and for supplying the 50 day history for the first day of the year.

II. Training Setup

A. Load dataset

Dataset is loaded from csv files.

B. Calculate additional features

The only additional feature calculated on the data is the day of the week 1-hot-encoded into 5 separate features.

C. Slice into training examples

The data then gets cut into time series, each with a length of 50. The data from the day following each time series is used to create the label, which consist of the relative increase in high price and low price compared to the close price of the previous day. The training examples from each stock get concatenated together into 1 dataset.

D. Normalize

Each training example gets all of it's price data normalized together on a scale of -1 to 1. Similarly, the volume data gets normalized on a scale of -1 to 1.

E. Separate validation set

Once all the training examples have been created, 15% of the training examples are sliced off of the back of the dataset to become the validation set. The rest of the examples is the training set.

F. Shuffle training data

The ordering of the training set gets shuffled.

G. Split batches

The training data gets split into whole batches of size 200.

H. Initialize model

The model is initialized with random parameters. The model is an LSTM with 1 hidden layer of size 64. The rest of the model architecture is depicted in "Fig. 1".

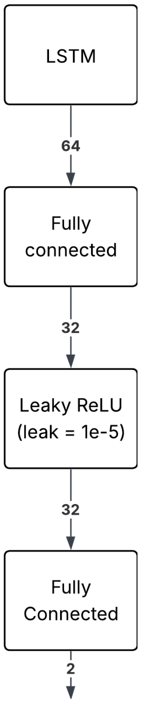

Fig. 1. Model structure.

III. Training Loop (per epoch)

A. Calculate loss

The loss for each batch is calculated with formula “(1)” . The formula utilizes Mean Squared Error loss and a magnitude penalty for the logits.

$$Loss = mse(Y, \hat{Y}) + \lambda \sum_{i=1}^{B} (\hat{Y}_{i,1}) - \lambda \sum_{i=1}^{B} (\hat{Y}_{i,2}) \tag{1}$$

Here the first logit is the high price that the model is predicting, and the second is the low price. B represents the batch size, and λ is the magnitude penalty coefficient.

B. Optimize

The loss is backpropagated for each batch, and the parameters are updated with the Adam optimizer. The learning rate for the optimizer starts at 1e-3, and is set by the PyTorch LowerOnPlateau scheduler. The metric tracked by the scheduler is validation loss.

C. Validation

After all the batches are used for training, validation occurs. The model predicts all of the examples in the validation set and loss is calculated. Unlike during training, the loss in validation is calculated as a pure Mean Squared Error loss with no magnitude penalties. The loss is then used to update the learning rate scheduler.

IV. Testing

A. Dataset setup

The testing set is loaded. The data undergoes feature calculation and normalization in the same manner as the training set.

B. Agent setup

An agent is given a trading balance of $100.

C. Get model predictions

The testing process iterates over each day in the years covered by the test set. For each day at timestep T, the model receives data from T_{-50} through T_{-1}, and predicts the high and low price at T relative to the close price at T_{-1}.

D. Calculate positions opened

To test the model, the agent employed a simple trading algorithm that aims to utilize strong signals and minimize losses. If the model predicts a high price increase $\geq 5\%$ and a low price increase $> -0.1\%$, the agent is considered to have purchased the stock at the close price for day T. The agent’s balance is evenly split between each of the stocks chosen by the model, with each getting a minimum of $1. If there are more stocks than can each be given $1, some will be removed based on how high their high price prediction is.

E. Calculate positions sell prices

For each position that the agent would have opened, it closes the position during the day at timestep T. The agent is attempting to sell the position for a price that is $\geq 5\%$ increase, but will stop-loss if the price is $\leq -0.1\%$. The outcome of the position is determined using the algorithm detailed in “Fig. 2”. The price that each position would’ve sold for is used to update the agent balance. Each position is checked first to see if the trading parameters would have caused it to sell at the open price. Then it is checked to see if it would sell during the day. Note the outcome where an a position value of both $\leq -0.1\%$ and $\geq 5\%$ occur during a day. Since it is ambiguous which would’ve happened first when using data at the level of 24 hours, it is considered a stop-loss in the interest of being conservative.

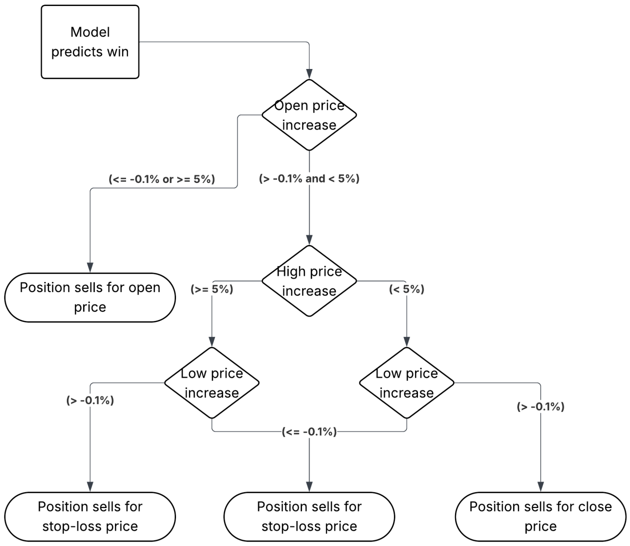

Fig. 2. Possible position outcomes.

F. Calculate positions outcomes

Each position that is sold at or above the aim of a 5% increase is counted as a ”win”. Each position that is sold at or below the stop-loss value of a -0.01% is considered a ”loss”. Any position that does not have either occur would be sold for the close price at the end of the day which was originally predicted. Such situations are considered a ”time-out”.

G. Calculate results

After process is repeated for every timestamp, the total balance increase is calculated for the whole time span, along with the rates at which each of the outcomes occur.