youyfin

Goal: create a neural network model that can automatically give signals for trading stocks.

Neural networks excel at finding statistical relationships between certain patterns. However, they don’t show any understanding of the things it is predicting. The fundamental idea that a neural network would work in this situation is that if a scenario that occurs today is very similar to one that occurred in the past, there will be a similar outcome.

Method:

Gathering Data:

Neural networks are very data hungry. In recent years they have grown massively in the number of parameters they have (some going up into the billions). If we are to examine daily data points, the number of parameters would be an order of magnitude greater than the data we have if we just choose a single stock over the past 20 years. So instead, we will gather data from large companies in an entire sector. This way we can prevent overfitting as well as having more data. Also, the data will be over the past 20 years since it may be a worry that the economic situation was so different back then that the patterns from those time is no longer that relevant. To start off we will be using the top 10 tech stocks by market cap in US. We will get the data from yahoo finance.

We will be splitting this data into 3 parts, training, validation and test. We will not use the test data until the very end to evaluate which model to use. The training and validation datasets will be continously used to improve the model.

Feature engineering:

Hand engineering features based on the raw data is a great way to implement human knowledge into neural network. For example, adding the exponential moving average can tell the neural net that this trendline is useful. On its own, the neural net might not be able realise the significance of these features.

Other than engineering features from the stock price and volume, we can also implement features that give information on the world economic state such as interest rates, government spendings etc.

One potential feature that we could implement could be sentiment analysis, analyzing news articles, social media posts to gauge the general public's opinion of a certain stock. This can be done using a separate neural network.

First model idea:

Training set 2010 – 2020, validation: 2021, test: 2022

21(shorten for testing purpose -> start with 3, need to tune this hyperparamter) days,

Jan 1 – jan 22 , pred: jan 23

Jan 2 – jan 23, pred: jan 24

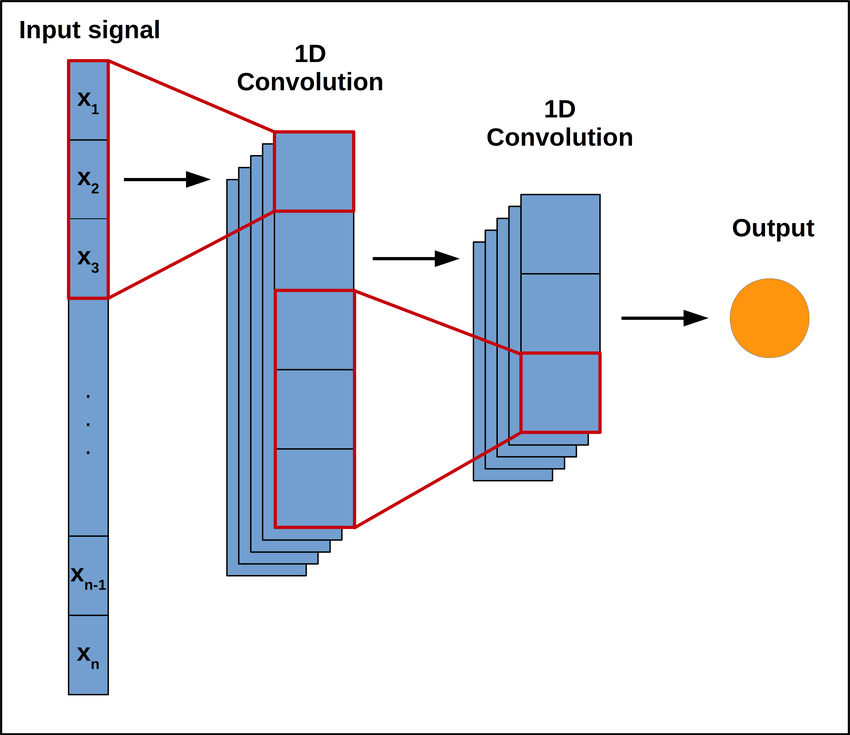
Jan 3 - jan 24 pred jan 25

CNN -> GRU -> neural net

We use some known trading indicators, stock price, and economic features discussed previously as a starting point for features. We turn the stock price into percentage increases and decreases (this is done as the model will become inaccurate in stock price ranges that it is not trained on) (we can further engineer this later, some ideas include cutting percentage below absolute value of 0.5% to get rid of noise and only mainly focus on bigger moves). Then we can have the structure of the model with the technical indicators feeding into the net like tabular data and having a convolutional neural network to first extract certain patterns from the stock data.

To explain the choice of a 1 dimensional CNN:

In a nutshell what a CNN does is that:



We take some sequential feature from the left and we take the dot product of a portion of that sequential feature with some preset vector we call the kernel.

For example for a sequence of :

1.6 -0.5 0.4 1.5 2

Let’s have some kernel: [1, 1.5, 2] . The reason such a kernel could arise is because if the stock price goes up 1% one day, 1.5% the next day and 2% the day after. This could mean tomorrow the stock is likely to go up again. So let’s take the dot product of this kernel with the first three terms:

[-1.6, -0.5, 0.4] . [ 1, 1.5, 2] = -1.55

Since those two are not very similar, the dot product is not very large. We put this value down as the first output of the sequence: [-1.55] Then we do the same thing moving forward to the next set of three sequence

[-0.5, 0.4, 1.5] . [ 1, 1.5, 2] = 3.1

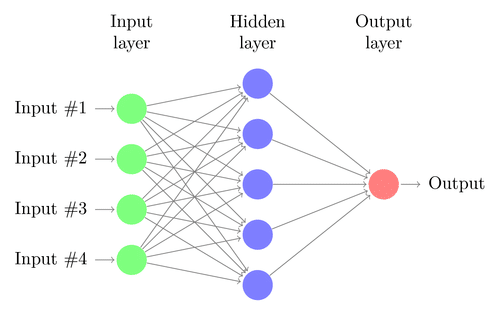
This time it is a bit more similar. We again add it to the output sequence: [-1.55, 3.1]

The next one [0.4, 1.5, 2] . [ 1, 1.5, 2] = 6.65

This time, their dot product is large since this is very close to the pattern we are looking for. So hopefully this feature that we have extracted give a big signal for the neural net that the price will most likely go up.

So the final output is [-1.55, 3.1, 6.65]

We can feed this into a deep fully connected layer concatenated with the other features. In the image below, input 1, 2 and 3 could be this output, input 4 can be two known useful technical indicators such as the moving average and MACD. Another input 5 could be the interest rates (not shown in the image). Then we let the neural network find some statistical correlation between these features and the result whether it will go up or down through some backpropagation algorithm.



Of course, we won’t be only using one such kernel, we can use many of them and they won’t be preset. In other words, the neural network can learn some very strange looking kernels that capture some very complicated non interpretable relationships.

Now, a fully connected deep layer doesn’t have to come immediately. We can use some sequential model such as LSTM (or transformer) to reduce a potentially a long list of sequence from the CNN into just a few features but those are ideas for the future.

Evaluating model:

We will evaluate the model first by mean squared error if we are predicting the percentage increase or accuracy if we are predicting the direction of the stock. However, that is not enough for a profitable trade. We will back test this on past data assuming we are trading futures for a more accurate evaluation of the model.