CS6140 Project 5

Haotian Shen
Qiaozhi Liu

Abstract

The objective of this project is to predict the direction of five stock markets (Dow Jones, NASDAQ, NYSE, Russell, and S&P) using machine learning techniques. The project is divided into four main tasks. The first task is to organize the data for 1-day binary prediction by creating a single CSV file that contains the data for each stock market for the same day, along with the next day's label (up or down). The second task is to implement two ML methods for 1-day binary prediction using either classic ML methods or deep neural networks. The third task is to implement an ML method that uses more than one day's worth of data to predict each market index, using a dataset created from the first task. Finally, the fourth task is to improve the model by iterating on the design to achieve better performance. This could include using more history, a more complex model, a different output definition, or feature extraction techniques such as PCA. The performance of the models will be evaluated based on their accuracy in predicting the direction of the stock markets.

Reflection

Through this project, I have learned several important concepts and techniques related to machine learning for financial forecasting. I have gained a better understanding of data preprocessing techniques such as data normalization, feature scaling, and data imputation. Additionally, I have learned about the different types of neural networks and how they can be applied to financial forecasting, including convolutional neural networks, recurrent neural networks, and feedforward neural networks. I have also gained a deeper appreciation for the importance of feature engineering and model selection in achieving accurate predictions. Through the process of iterating on different models and feature sets, I have seen how small changes can have a significant impact on the model's performance. This has reinforced the importance of an iterative approach to machine learning and the need to constantly evaluate and improve on models.

Task1: Organize your data for 1-day binary prediction

Instead of 2 labels, we decided to use 3 labels, UP, DOWN, and FLAT since we noticed that some of the closing prices are the same as their previous day’s price. The ‘Close’ column was renamed to ‘[MARKET_NAME]_price’ and its price direction column was named ‘[MARKET_NAME]_price_dir’. The script is named task1.py and the output can be found in the CNNpred folder called day1prediction.csv. This data file was then split into training and test sets named ‘day1prediction_train.csv’ and ‘day1prediction_test.csv’.

Task2: Implement two ML methods for 1-day binary prediction

We will preprocess the data to prepare for X_train, X_test, y_train, y_test. To prepare for X data, drop the columns ending with price and price_dir to only keep the technical analysis features. To prepare for y data, just use the columns ending with price_dir. We will use the same model to predict all 5 markets’ price direction.

Instead of 2 ML methods, we decided to utilize 3 ML methods for a more comprehensive prediction for the stock dataset. Each model was trained for 10 epochs during this task.

1. **Feedforward neural network:**
    a. **Design decision**: FFNN is a type of ANN that is suitable for time series prediction. It can be used for stock price prediction because it can learn to model the complex, nonlinear relationships between the input features and the target output. FFNN can process large amounts of data to extract relevant patterns and relationships. It can also generalize from training data to predict unseen test data to be able to make accurate predictions on future using stochastic gradient descent.
    b. **ML system**: It is a simple FFNN model that takes in an input tensor of shape (batch_size, input_size) where input_size is 83. It consists of three fully connected layers with 64, 32, and output_size number of neurons, respectively. The output_size of the model depends on the number of classes we are trying to predict. The first two fully connected layers use the ReLU activation function, while the last layer is a linear layer. The input tensor is first flattened using the view() function with the batch size remaining unchanged and the remaining dimensions flattened. The model applies a ReLU activation function to the outputs of the first two layers and returns the final output from the last layer.
    c. Performance:

market	accuracy
DJI	0.560538
NASDAQ	0.385650
NYSE	0.452915
RUSSELL	0.542601
S&P	0.573991

Average price direction prediction accuracy: 0.5031390134529148

2. **Recurrent neural network**:
    a. **Design decision**: RNN is suitable for stock prediction because it is naturally used for processing sequential data. Technical analysis features such as moving averages, relative strength index (RSI), and moving average convergence divergence (MACD) rely on past price information to calculate their values. RNNs can capture these temporal dependencies. It can also remember information from the past, allowing them to learn from historical data and capture long-term trends in stock prices.
    b. **ML system**: The input_size is 83, which corresponds to the number of technical analysis features. The RNN layer takes in the input and produces a hidden state with hidden_size, which is then passed to the fully connected layer (fc) that produces the output_size. The batch_first parameter is set to True, which means the input is expected to have the shape (batch_size, sequence_length, input_size). In the forward pass, the initial hidden state (h0) is initialized with zeros and has the shape (1, batch_size, hidden_size). The input (x) and the hidden state (h0) are then passed to the RNN layer, which returns the output tensor (x) and the final hidden state (hn). Finally, the output tensor is passed to the fully connected layer, and the output of the last time step (-1) is used as the prediction for the direction of the stock price.
    c. Performance:

market	accuracy
DJI	0.560538

NASDAQ	0.385650
NYSE	0.547085
RUSSELL	0.457399
S&P	0.573991

Average price direction prediction accuracy: 0.5049327354260089

3. **Long short-term memory neural network:**
    a. **Design decision**: LSTM can capture the sequential patterns in the stock dataset since technical analysis features are often time-series data. It can model dependencies and retain memory of past inputs. In stock price prediction, it is often important to consider the historical patterns in the data, as the current stock price may be influenced by past trends and market conditions. LSTM networks are able to learn and remember these historical patterns and use them to make predictions about future price direction.
    b. **ML system**: This LSTM neural network takes in an input size of 83, which represents the number of technical analysis features used as input. The network consists of an LSTM layer with a hidden size of 'hidden_size' and 'num_layers' number of layers. The LSTM layer takes in the input and generates output at each time step, followed by a fully connected linear layer with 'output_size' number of nodes to generate the final output. The initial hidden state and cell state are both set to zero using 'h0' and 'c0'. The output of the LSTM layer is passed through the fully connected layer, and the final output is obtained from the last time step using 'x[:, -1, :]'.
    c. Performance:

market	accuracy
DJI	0.560538
NASDAQ	0.614350
NYSE	0.452915
RUSSELL	0.542601
S&P	0.573991

Average price direction prediction accuracy: 0.5488789237668161

Analysis

Based on the results of the price direction predictions using the three different types of neural networks, we can see that the performance varies across different markets. The accuracy for predicting the direction of the NASDAQ using the FFNN is only 38.6%, while the accuracy for predicting the same for the NYSE using the same model is 54.7%. The LSTM seems to perform better than the FFNN and RNN models for most of the markets, with the highest accuracy being achieved for the NASDAQ at 61.4%.

Overall, the LSTM model seems to have the highest accuracy across all markets, although the accuracy is still not very high. It is also worth noting that the performance of the models could be improved by adjusting the hyperparameters or trying different models altogether. We will use this as the baseline ML result and experiment with CNN in the next task with the hope to improve the prediction accuracy on the test data.

Task3: Implement an ML method that uses more than 1 day to predict each market index

We used CNN for this task to align the ML method with the one utilized within the CNNpred paper. We created the dataset on the fly using the day1prediction.csv file and just use 2-day worth of day to generate our baseline CNN result with the goal to improve the accuracy in the next task. The CNN model was trained for 10 epochs as well in this task.

Design decision:

CNN neural network is suitable for stock price direction prediction based on technical analysis features because CNN can effectively capture local patterns and relationships between technical features across multiple time frames. Technical analysis is concerned with identifying patterns in stock price and volume charts, and these patterns can be represented as local features that are relevant to a particular time frame. By using convolutional layers, CNNs can learn to identify these local features and their relationships across multiple time frames. This allows the network to learn more complex patterns in the data that may not be obvious from looking at individual technical features. Additionally, the pooling layers can help reduce the dimensionality of the data, making it easier for the network to learn and generalize to new data.

ML system:

This CNN network is designed to take in input features with 83 channels (i.e., technical analysis features). The first convolutional layer (self.conv1) has 60 output channels and a kernel size of 3. The second convolutional layer (self.conv2) has 30 output channels (which is the result of dividing 60 by 2, as specified by conv1_out_channels // 2) and also has a kernel size of 3. The output of the second convolutional layer is then flattened and passed through two fully connected layers. The first fully connected layer (self.fc1) has 60 output channels and takes in input channels that are the result of time_steps // 2 * conv1_out_channels // 2. The second fully connected layer (self.fc2) has num_classes output channels, which corresponds to the number of classes in the prediction task. During the forward pass, the input tensor x is permuted to have the channel dimension as the second dimension. This is necessary because the Conv1d layer in PyTorch expects the channel dimension to be the second dimension. The resulting tensor is then passed through the two convolutional layers with ReLU activation function, followed by a max pooling layer with kernel size of 2. After that, the tensor is flattened and passed through two fully connected layers with ReLU activation function (except for the last layer), and the output is returned as a tensor with num_classes channels.

The input dataset creation method takes as input the dataset features and the target variable along with a window size which defines the number of days to use as input to predict the target variable. The method then loops through the input dataset and extracts input windows with a size equal to the specified window size. The extracted windows are then appended to a list of inputs. The target variable for the corresponding window is also extracted and appended to a list of targets. The method then returns two arrays, one containing all the extracted input windows, and the other containing all the corresponding target variables. In the case of stock price direction prediction using 2-day worth of data, the window size would be set to 2, and each input window would contain the technical analysis features for two consecutive days. The method would extract all possible 2-day windows from the dataset and append them to a list of inputs along with their corresponding stock price direction labels, which would be the target variable. The created dataset is then passed to the model as a 3-D input tensor.

Performance:

market	accuracy
DJI	0.565611
NASDAQ	0.380090
NYSE	0.547511
RUSSELL	0.542986
S&P	0.579186

Average price direction prediction accuracy: 0.5230769230769231

Analysis

Looking at the accuracy results, the CNN model is performing relatively well on predicting the stock direction for the Dow Jones Industrial Average (DJI), NYSE, and S&P, with an accuracy rate of around 54%-57%. However, it performs relatively poorly on the NASDAQ and RUSSELL indices, with an accuracy rate of only around 38% and 54%, respectively.

The relatively poor performance on NASDAQ and RUSSELL indices may be due to several factors, including:

1. Number of epochs are not enough: The training epochs was only 10 epochs, which might not be enough to achieve convergence on the model. Increasing the number of epochs may yield better results.
2. Lack of relevant features: The technical analysis features used to predict the stock direction may not be relevant enough to predict the stock direction of the NASDAQ and RUSSELL indices. Different indices may have different features that are more relevant in predicting their direction.

Task4: Improve your model

Improvements:

1. Increase epochs from 10 to 30.
2. Increase the time steps from 2 to 30 to use more days worth of data for prediction.
3. Define the output differently using more fine-grained classification.
4. Use a more complex model by increasing the output channel of the first convolution layer from 60 to 160.

Design decision: Using UP, FLAT, and DOWN eliminates the potential fine-grained relationships within our dataset, so we classified the data differently on the target labels. Increasing the number of training epochs can also help the model to better fit on the training data to prevent underfitting. Using a more complex model can help the model learn more complex relationships between features. Using more days of previous features can also improve the performance by preserving the relationship in time series data and help the model by giving it a temporary memory context.

ML system: Use SMALL_UP, BIG_UP, FLAT, SMALL_DOWN, BIG_DOWN. The demarcation of small and big happens at 1.5% percentage change of the current price compared to the previous day's

price. We also trained with more epochs to help the ML model to converge more on the dataset. Using a larger output channel size renders a more complex model, so it can perform the expansion and contraction of the size of weights connected to different layers to help the model learn more complicated and sophisticated relationships between different technical analysis features. These changes could potentially improve the accuracy of the test set.

Performance: This performance is derived by combining all 4 improvements

market	accuracy
DJI	0.580311
NASDAQ	0.601036
NYSE	0.549223
RUSSELL	0.528497
S&P	0.580311

Average price direction prediction accuracy: 0.5678756476683937

Other improvements:

1. Normalize the dataset.
2. Perform PCA on features over the entire dataset to create a projected dataset with salient features that preserve 95% of the original variations.

Design decision: Previously, we always trained the network model using our unnormalized dataset, and this means the dataset is not in a uniform unit. This could potentially hindered the performance of the model since the model needs to learn the relationships of different units, which is not really necessary. Predicting the accuracy of the stock market is also a very sophisticated task that generally does not yield promising results, so if we could not improve the accuracy anymore, we can lower the dimensionality of the stock technical analysis features and improve the model performance by achieving similar test accuracy with less computational cost.

ML system: Normalize the dataset and perform PCA on all features to generate a projected X_train and X_test dataset. The input tensor will be generated with the lowered dimensionality.

Performance:

It turned out that normalizing the dataset does not even change the accuracy. The same results were obtained after just normalizing.

market	accuracy
DJI	0.580311
NASDAQ	0.601036
NYSE	0.549223
RUSSELL	0.528497
S&P	0.580311

Average price direction prediction accuracy: 0.5678756476683937

Applying a PCA with 95% kept variation reduced the dimensions from 83 to just 37.

market	accuracy
DJI	0.414508
NASDAQ	0.601036
NYSE	0.549223
RUSSELL	0.528497
S&P	0.580311

Average price direction prediction accuracy: 0.5678756476683937

Surprisingly, the test accuracy of the DJI market has decreased significantly. This could potentially mean that the eliminated features are important in predicting DJI market price, but are not affecting the performance of other market prices. However, with only 37 features, we achieved similar test accuracy compared to using 83 features, which is a significant reduction in computational cost, so the result is very acceptable.

Extension

We have done multiple improvements and iterations to improve the model and also tried some extra models to compare their performance.

1. Used LSTM model for stock prediction with memory context.
2. Apart from the required 1 iteration of improvement, we conducted 5 more iterations of improvement to generate our best performing model with average test accuracy on all 5 markets as 0.5678, and the highest on a single market was 0.601036 on NASDAQ.

Acknowledgement

No external consultation was made.