Investing in SPY ETF: Using ML on SPY Constituents' Momentum Data

Project Member – Dale Angus
Project Category – Stock Market Investing

Proposal

In this project, I would like to determine if price and price momentum and other pieces of related data could be a reliable input for a machine learning model that would predict, in the short-term, the price direction of the SPY ETF.

Models

Model A

I used an L-layer Classifier model based on the "Deep Neural Network Application Image Classification" taught in class (week4). I used most of the code provided and I changed a few items such as the *loaddata()* function, *layer_dims* and *learning_rate* variables. I also removed some code that is specific to the image classification application.

Model B

I used an 8-class Softmax Classifier model based on the "Tensorflow Tutorial". Like Model A, I changed the *loaddata()* function and changed some parameters to fit my application.

Dataset and Features

The dataset is composed of **two years of daily** prices and related information of the **499 constituents** of the SPY ETF. Two years of historical price data per constituent would have 470 rows; multiplied by 499 should be around 234000 rows. Below are the features (columns) of the constituents' data that were used.

`sma10, sma21, slope10, slope21, awesomeoscillator, momentum34, tangentslope, calcdate, ticker, open, high, low, pctchange, weighting, sector`

In addition, information of the related ETFs, i.e. **VIX**, **DIA**, **IWM** and **ONEQ**, are included and linked (or joined) to the above data using the closing date. The columns are,

`vixclose, vixdataid, vixticker, vixpctchange, diaclose, diadataid, diaticker, diapctchange, iwmclose, iwmdataid, iwmticker, iwmpctchange, oneqclose, oneqdataid, oneqticker, oneqpctchange`

For **Model A**, the classification label is the sign of the percentage change value of the SPY ETF. It is represented as **0 for negative** and **1 for positive**. The column is `spyupdown`.

For **Model B**, the eight classes are the SPY ETF end-of-day percentage change in 1% buckets as follows:

$$\begin{bmatrix} \Delta \leq -3\% \\ -3\% < \Delta \leq -2\% \\ -2\% < \Delta \leq -1\% \\ -1\% < \Delta \leq 0\% \\ 0\% < \Delta \leq 1\% \\ 1\% < \Delta \leq 2\% \\ 2\% < \Delta \leq 3\% \\ 3\% < \Delta \end{bmatrix}$$

The columns are `dn3, dn2, dn1, dn0, up0, up1, up2, up3.`

Features and Labels

1. Weighting, Sector – The weights used for each constituent are not the true historical weights but the respective current weights **on the day the dataset was collected***. The sector to which the constituent belong.
2. Calcdate (or Close date), Open*, High*, Low*, Close, Volume, Percent Change – Daily closing information
3. SMA10* and SMA20* – Simple 10-day and 20-day moving averages
4. Slope10* and Slope20* – Slope of the linear regression line of the 10 and 20-day prices
5. Awesome Oscillator* and 34-day Momentum* – Momentum indicator calculated using their respective formulas
6. Tangent Slope* – Calculated by getting the slope of the tangent line at the last data point of the the 2nd-power polynomial (Least Squares Method) equation using 21 data points*
7. [ETF]pctchange – This is the ratio between the constituent's percent change and the other related ETF's percent change. The other related ETFs are **DIA**, **QQQ**, **IWM,** the index **VIX. DIA, QQQ, IWM** were chosen because they represent the most-followed market benchmarks. **VIX** was chosen because it is the benchmark measure of the volatility of the SPY.
8. Spyupdown – the label that represents whether the price of the SPY ETF went up (1) or down (0)
9. DNx and UPx – these labels represent the 8 softmax classifications.

The columns with asterisks were divided by the constituent's price. The idea is to eliminate the big difference between the constituents' prices, for example, Amazon.com (AMZN) has a price of $1581.76 and Ford Motor Co. has a price of $11.46.

**A special note about weighting*: In the absence of the historical data of weightings of each constituent, I used the most current weighting at the time the data was gathered (2018-05-18). The idea is that, the actual weight of each constituent could not have changed that much within the two year period. For example, Apple would have maintained its highest weighting in the ETF within the two year period.

Source Code

A copy of the source code is in GitHub.

Model A
https://github.com/daleangus/proof_im_a_developer/blob/master/finance.py
Model B
https://github.com/daleangus/proof_im_a_developer/blob/master/finance_softmax.py
Java program that shows the calculations of the momentum-related values
https://github.com/daleangus/proof_im_a_developer/blob/master/CalculatedPricingData.java

The rest of the code used by my model that is not posted in GitHub is from the same image classification application stated above. It is in the *dnn_app_utils_v3.py* file for Model A and *tf_utils.py* for Model B.

Two things that I want to point out,

1. The NULLs in the raw dataset is replaced by the column's mean.

```
fill_value = pd.DataFrame({col: traindata.mean(axis=1) for col in
traindata.columns})
traindata.fillna(fill_value, inplace=True)
```

2. The sector dictionary is used to represent the sector names as numbers.

```
sector_dictionary = {'Basic Materials':1, 'Capital Goods':2,
'Conglomerates':3, 'Consumer Cyclical':4, 'Consumer/Non-Cyclical':5,
'Energy':6, 'Financial':7, 'Healthcare':8,'Services':9, 'Technology':10,
'Transportation':11, 'Utilities':12}

traindata['sector'] = traindata['sector'].apply(lambda x:
sector_dictionary[x])
```

Saved Parameters

The trained parameters are saved to disk using the **Pickle** Python package. Being able to save the parameters allowed me to use the trained parameters in **production**. The code snippet below shows how to save and restore the **trained parameters**.

```
import pickle

#save
timestr = time.strftime("%Y%m%d-%H%M%S")
print("Saved as " + "tf_softmax_nn-" + timestr + ".pickle")
pickle_out = open("tf_softmax_nn-" + timestr + ".pickle", "wb")
pickle.dump(parameters, pickle_out)

#restore
pickle_name = 'tf_softmax_nn-20180530-194327.pickle'
with open(pickle_name, 'rb') as handle:
    parameters = pickle.load(handle)
```

Training and Test

Hyperparameter Search

With **Model A**, I initially limited the number of iterations to 5000 because I was too eager to see the results. Later I realized that I had to run the training for bit longer. Since I don't want to waste time waiting for one result to finish, I configured 5 other computers to train simultaneously. First, using the same neural network layers, I studied the effect of the learning rates on the accuracy. Once satisfied with the learning rate value, I went ahead to test different neural network configurations. I ran the model up to 100,000 iterations or until the absolute difference between the consecutive costs is less than 7×10^{-6}. With Model A, many times, the backward propagation step results in *NaN* error especially when running longer iterations. I also noticed that adding more layers doesn't necessarily generate better accuracy. With **Model B**, I tried to see if I can improve the softmax prediction by testing different learning rates, epochs and mini-batch sizes. I used the Adam Optimizer for Model B.

Both models use Xavier initialization and the cost function,

$$-\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}\log\sigma\left(z^{|L|(i)}\right)+\left(1-y^{(i)}\right)\log\left(1-\sigma\left(z^{|L|(i)}\right)\right)\right)$$

Features Selection

For both models, I focused on studying the effects of including or excluding the **weightings** and the data **related to other ETFs**.

Results

The table shows the stages of the development of the models and production prediction accuracy. Below are the different test results.

	Accuracy	Accuracy	Accuracy	Accuracy
Features	**Train 56.42% |Test 56.41%**	**Train 56.47% |Test 55.97%**	**Train 57.2% |Test 57%**	**Train 59.67% |Test 59.78%**
Percent Change	✓	✓	✓	✓
Slope10	✓	✓	✓	✓
Slope20	✓	✓	✓	✓
Awesome Oscillator	✓	✓	✓	✓
Momentum	✓	✓	✓	✓
Tangent Slope	✓	✓	✓	✓
Open	✓	✓	✓	✓
High	✓	✓	✓	✓
Low	✓	✓	✓	✓
Weighting		✓		✓
VIX Close	✓	✓	✓	✓
VIX Pct Change	✓	✓	✓	✓
Sector	✓	✓	✓	✓
DIA % Change ratio			✓	✓
IWM % Change ratio			✓	✓
ONEQ % Change ratio			✓	✓
	layer_dims: [12, 10, 9, 5, 4, 1]	layer_dims: [12, 10, 9, 5, 4, 1]	layer_dims: [12, 10, 9, 5, 4, 1]	layer_dims: [12, 10, 9, 5, 4, 1]
	learning rate: 0.0075	learning rate: 0.0075	learning rate: 0.0075	learning rate: 0.0075
	iterations: 2000	iterations: 2000	iterations: 2000	iterations: 2000
Comment	**BASE**	**LITTLE IMPROVEMENT**	**BETTER**	**BEST**

Table 1a: **Model A** Study the effects of weighting and the related ETF data. Getting a very high accuracy is not the goal in these tests. The goal is to inspect the effects of including or not the said features.

	Accuracy	Accuracy	Accuracy	Accuracy
Features	**Train 78.5% |Test 73.32%**	**Train 79.59% |Test 79.14%**	**Train 97.74.2% |Test 97.74%**	**Train 98.3% |Test 98.1%**
Percent Change	✓	✓	✓	✓
Slope10	✓	✓	✓	✓
Slope20	✓	✓	✓	✓
Awesome Oscillator	✓	✓	✓	✓
Momentum	✓	✓	✓	✓
Tangent Slope	✓	✓	✓	✓
Open	✓	✓	✓	✓
High	✓	✓	✓	✓
Low	✓	✓	✓	✓
Weighting		✓		✓
VIX Close	✓	✓	✓	✓
VIX Pct Change	✓	✓	✓	✓
Sector	✓	✓	✓	✓
DIA % Change ratio			✓	✓
IWM % Change ratio			✓	✓
ONEQ % Change ratio			✓	✓
	layer_dims: [12, 25, 12, 8]	layer_dims: [13, 25, 12, 8]	layer_dims: [15, 25, 12, 8]	layer_dims: [16, 25, 12, 8]
	learning rate: 0.0001	learning rate: 0.0001	learning rate: 0.0001	learning rate: 0.0001
	epoch: 3000	epoch: 3000	epoch: 3000	epoch: 3000
	minibatch: 32	minibatch: 32	minibatch: 32	minibatch: 32
Comment	**BASE**	**LITTLE IMPROVEMENT**	**BETTER**	**BEST**

Table 1b: **Model B** Study the effects of weighting and the related ETF data. Getting a very high accuracy is not the goal in these tests. The goal is to inspect the effects of including or not the said features.

Production Results

One row (sample) represents the day's data of **one** constituent. The **significance of this is that even one sample**, and not the 499 constituent data altogether, can be used to predict. But why use one sample when there are up to 499 available samples to test in production. The training/test data is from **2016-07-07** to **2016-05-18.**

Market Date	SPY % change	Ground Truth - 0 or 1	Out of 499 samples
21-May	0.75%	1	100.00%
22-May	-0.28%	0	88.18%
23-May	0.28%	1	100.00%
24-May	-0.20%	0	100.00%
25-May	-0.24%	0	99.79%
29-May	-1.15%	0	100.00%
30-May	1.33%	1	100.00%
31-May	-0.61%	0	96.79%

Table 2a: **Model A** Production Predictions. Saved parameters file, *dnn-20180527-220443.pickle*

Market Date	SPY % change	Ground Truth - Ordinal	Out of 499 samples
21-May	0.75%	4 ($0\% < \Delta \leq 1\%$)	100.00%
22-May	-0.28%	3 ($-1\% < \Delta \leq 0\%$)	100.00%
23-May	0.28%	4	100.00%
24-May	-0.20%	3	0.00%
25-May	-0.24%	3	95.59%
29-May	-1.15%	2 ($-2\% < \Delta \leq -1\%$)	3.41%
30-May	1.33%	5 ($1\% < \Delta \leq 2\%$)	98.60%
31-May	-0.61%	3	100.00%

Table 2b: **Model B** Production Predictions. See Model B Softmax classes under Dataset and Features. Saved parameters file, *tf_softmax_nn-20180610-134612.pickle*

Possible Application

The model predicts **the end-of-day** SPY ETF price direction using data **for that same day**. Obviously, if one would like to use the prediction of this model to trade in the stock market, he/she cannot wait for the end-of-day data (market is already close!).

So, in the **both** model's current form, the best way to use it is by using real-time data. At a certain point in time during market hours, e.g. 8:30 am PT when the European Market closes or 5 minutes before the close of market, a snap shot of that time's prices and other calculations could be used as estimated substitute for **what could be the** end-of-day values. All the input data from items 1 to 7 from the features listing above can be gathered.

Future improvements of the model would include predicting SPY ETF direction for next day, 3rd day, 4th day and so on. In addition, predictions of other ETFs could be made.

References

Historical Data, Yahoo Finance. https://finance.yahoo.com/

SPY ETF and Constituents, https://us.spdrs.com/en/etf/spdr-sp-500-etf-SPY?fundSeoName=spdr-sp-500-etf-SPY

Awesome Oscillator, https://www.metatrader5.com/en/terminal/help/indicators/bw_indicators/awesome

Momentum Oscillator, https://www.metatrader5.com/en/terminal/help/indicators/oscillators/momentum

ETF Correlation, http://www.quantf.com/ETF-correlations.php