Ensemble Reinforcement Learning for Futures Trading

Summary

Three reinforcement learning algorithms were used to create an ensemble trading strategy: A2C, PPO, and DDPG models. These models were shown the closing price, MACD, RSI, CCI, and ADX for 78 futures and expected to devise policies to maximize trading rewards. For risk management, a turbulence threshold was set at the 95th percentile of turbulence from the in-sample period.

The strategy achieved an annualized Sharpe ratio of 1.006 ($\mu_{r,daily} = 0.00056, \sigma_{r,daily} = 0.0088$) over the last 500 trading days (approximately two years) but has room for significant improvement. Scaling the observations and the rewards during training led to improved performance on the trading data, but overfitting severely reduced profitability on the validation and test sets.

Plans to improve strategy performance: further tuning of hyperparameters, such as learning rate and number of training epochs; adjusting technical indicators in the observation space; limiting the number of tradable contracts to improve tractability in the action space; implementing further risk management, such as stop-loss orders or allocation limits; train models on significantly more data, possibly synthetized by GANs.

Methodology

Data and Exploration

The data comes from Quantiacs, an online platform for algorithmic trading competitions. The data contains OHLCV price data for 78 futures contracts, along with rollover and open interest data. While the data goes all the way back to 1990, not all contracts have been available that long. Missing values were forward-filled to avoid look-ahead bias, and rows which could not be filled this way were dropped. The data was then cropped to the period between 2009-01-01 and 2020-12-01. No missing values remained. The end date was selected to avoid an error in the data in December 2020.

2009-01-01 through 2015-01-01 was selected as the in-sample period. The only use of the in-sample period was to determine a turbulence threshold for risk management. Financial turbulence, defined by Kritzman and Li (2010), can be thought of as a quantitative measure of the distance between the correlation structure of the market and previous correlation structure of the market; that is, a measure of price movements that deviate from the recent norm. The 90th percentile of the in-sample turbulence was used as the turbulence threshold for trading. If the turbulence in the trading period is greater than the threshold, then the algorithm simply closes all positions. This was helpful in the especially turbulent period in early 2020, when the Coronavirus pandemic caused swift market crashes.

The in-sample period was explored for correlation between close prices. As expected, there was significant positive correlation between many futures, but there was also negative correlation between a small number of pairs, and near-zero correlation was also common. This should allow effective diversification, at least in the framework of Modern Portfolio Theory.

The technical indicators (Moving Average Convergence-Divergence, Relative Strength Index, Commodity Channel Index, and Average Directional Index) were calculated using the StockStats package. For the RSI, CCI, and ADX were calculated using a 30-day rolling window.

Reinforcement Learning Models

The three models selected for the ensemble trading system were Advantage Actor Critic (A2C), Promixal Policy Optimization (PPO), and Deep Deterministic Policy Gradient (DDPG). All three are available in generic form in the Stable Baselines package created by OpenAI.

The models are all actor-critic algorithms, in which there are separate function-approximators that optimize actions and rewards. Thus, they allow for continuous actions spaces as well as continuous observation spaces.

Other algorithms were considered but ultimately discarded due to the relative inefficiency and comparable performance compared to the three selected algorithms.

Environments

Three custom environments were created: one for training, one for validation, and one for testing. The environments are quite similar.

The action space is vector of length 78, and continuous in [-1,1]. The action values indicate how many of each contract to buy/sell. Negative values indicate sales, while positive values are buy signals. The value between [-1,1] is multiplied by 100; a single transaction at most buys/sells 100 of a single contract.

The observation space is a vector of length 469. The algorithms can "see":

- Cash balance in the account [1]
- Close prices for each future [78]
- Shares already owned [78]
- MACD for each contract [78]
- RSI for each contract [78]
- CCI for each contract [78]
- ADX for each contract [78]

At each daily step, the algorithm takes in the 469 values in the state, and either predicts the best action (in validation/testing) or explores possible actions (in training). The actor then buys/sells the predicted number of contracts. The following day, the value of the portfolio is calculated using the new contract prices. The difference in value from the previous day is the reward. In the validation and trading environments, the daily differences are used to calculate Sharpe ratios. At a terminal state, the portfolio value is calculated but the next reward is zero.

System

Trading begins with a 9-month training period. The system is the recursive, but each training period begins with 2009-01-01. The steps are as follows.

1. Each algorithm is trained on the states from the training period.
2. Each trained model is validated on the next three months of data, and the Sharpe ratio for each model over the three-month validation window is calculated.

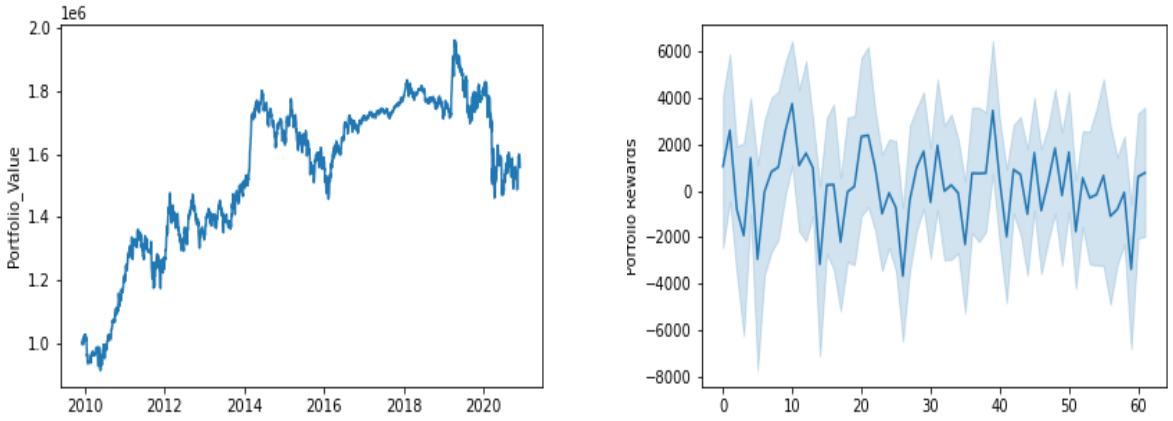
3. The model with the highest Sharpe ratio is then used to trade (in the testing environment) for the next three months. The results of the testing period are saved.
4. The validation window is then added to the training period, the testing window becomes the new validation window, and the cycle repeats.

Results

The system begins with $1,000,000 in cash. Several different versions, with small tweaks (mostly to hyperparameters) accounting for the difference. All versions of the system were plagued with poor performance at the beginning of the out-of-sample period, in 2015, but this poor performance could not be verified as connected to changes in turbulence. Commonly, the system performed poorly in 2019 as well. The cause of this second period of net losses is also undetermined.

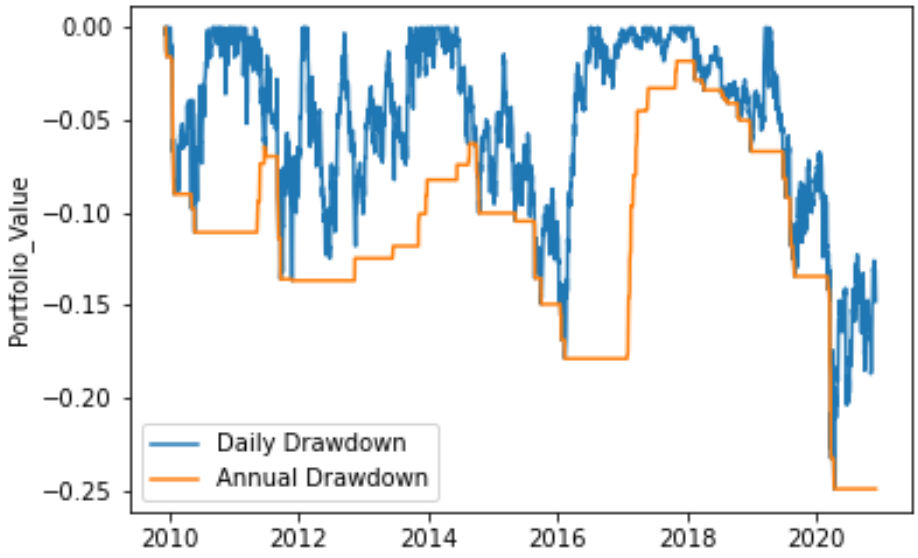
Despite these two periods wherein the system often faced losses, there were also two periods in which the system typically performed quite well; the time leading up to the setbacks in 2015, and the time from 2016 through 2018.

Both of these behaviors, both positive and negative, are visible in the following plot of total portfolio value. The plot on the right shows that the algorithms struggled to maximize rewards.



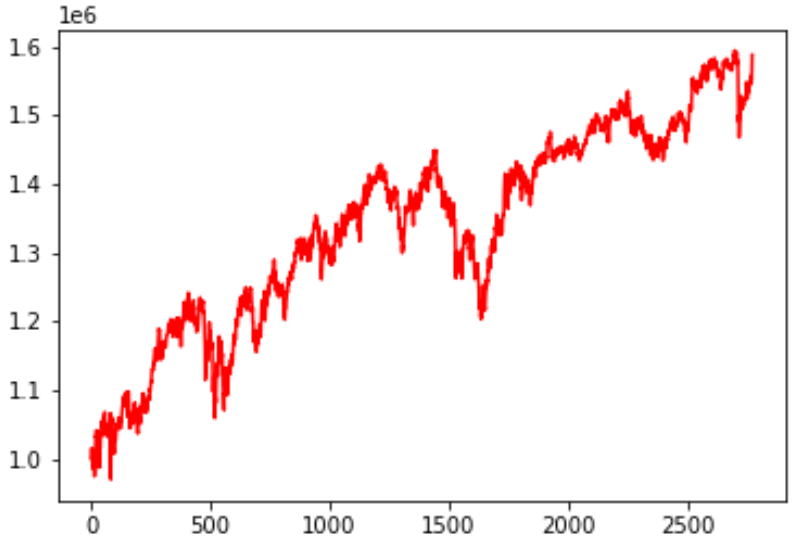
The strong negative performance in 2020 in this version coincides with the market crash caused by the coronavirus pandemic.

The drawdown plot shows the daily and annual drawdown; the system has a major drawdown in late 2014, but creates a new peak before falling to the maximum drawdown of approximately 25% in early 2020. This version was profitable on 50.2% of days, with a mean-win/mean-loss ratio of 1.05:1, indicating expected profitability.

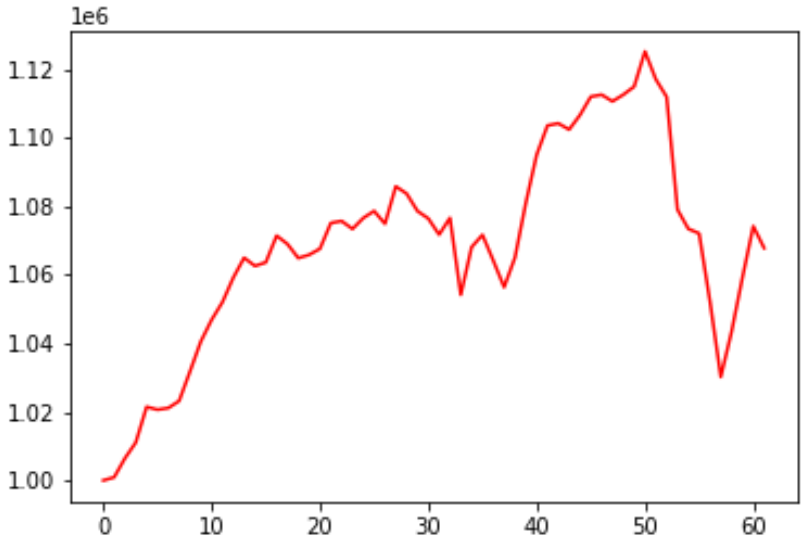


The results presented here were selected because they show good traction in the training performance while maintaining decent performance in the validation data. Other versions of the system showed better performance in training but exceedingly poor performance in validation, indicative of overfitting on the training data.

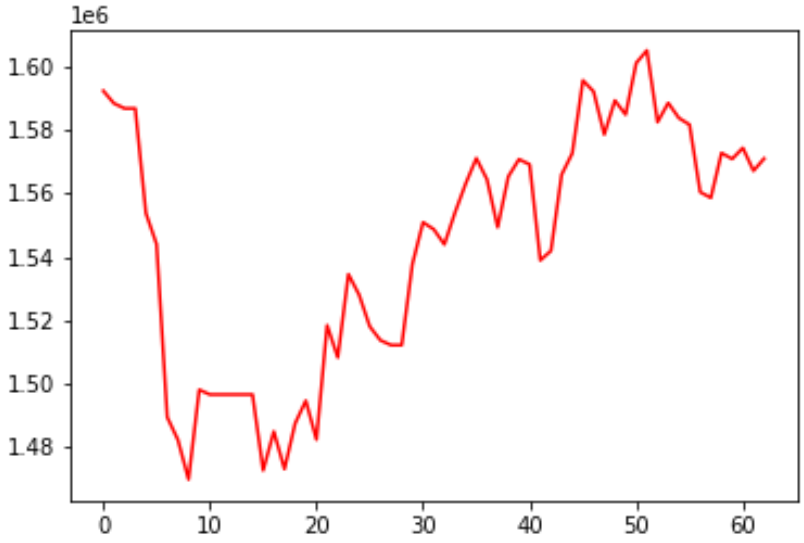
The following plot of performance in the training data shows that the algorithms are in fact learning useful policies.



The next plot shows similar performance in the validation window.



Then, we can see similar performance once again in the trading window.



This performance, though not stellar in financial terms, indicates that the underlying ideas behind the system have merit, and that performance could probably be improved with a combination of feature engineering, hyperparameter tuning, and regularization.

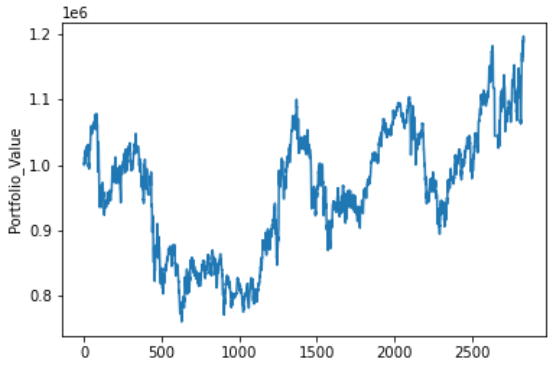
Alternate Results

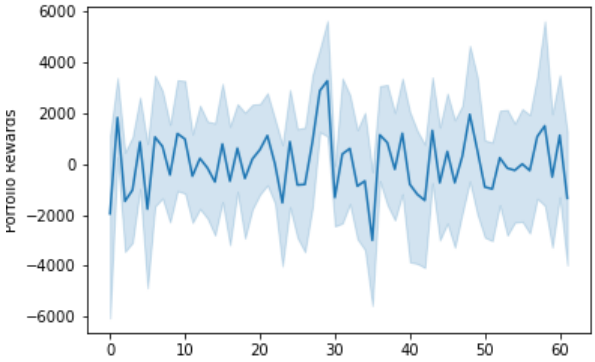
For comparison, we present the results from the most recent iteration of the system. In this version, the inputs are scaled:

- Close prices are divided by the starting close price on 2009-01-01
- RSI and ADX are divided by 100, scaling them in [0,1]

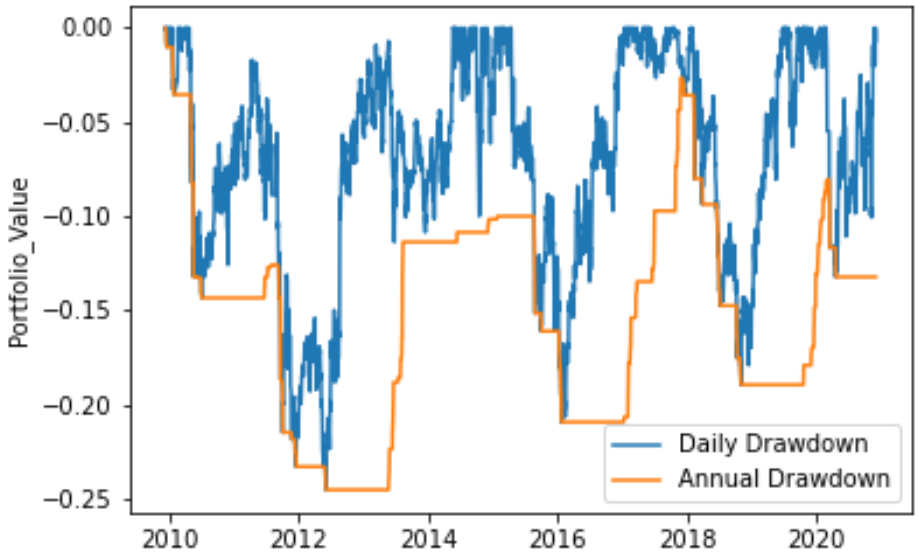
- CCI is standardized; centered at 0 with standard deviation 1
- MACD is left along, since it is centered at zero and generally between -1 and +1

Final results for this version are in the plot below. Notice wild swings in account value. Notice in the plot on the right that this version also struggled to predict rewards, and in fact has a negative mean reward.





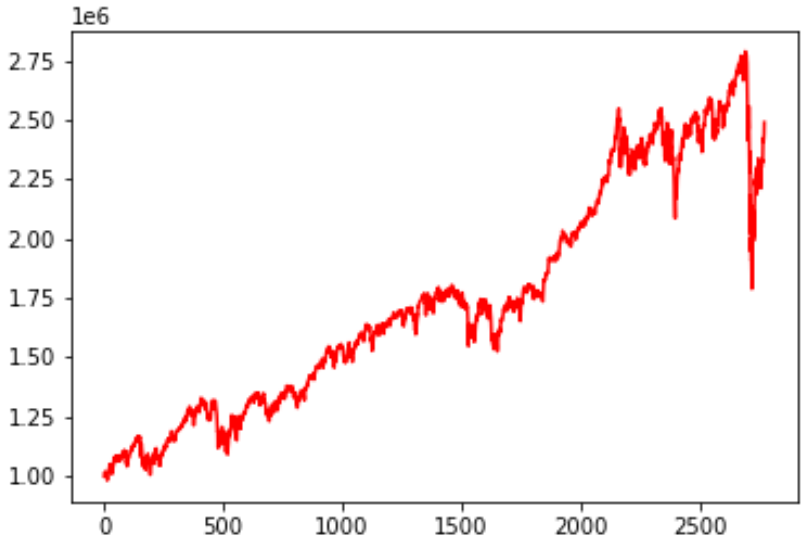
The drawdown plot for this version shows a similar maximum drawdown to the previous version, around 25%. This version however ends with a lower drawdown, since it outperformed the previous, unscaled model in 2020. This version was profitable on 49% of days, with a mean-win/mean-loss ratio of 1.03:1, indicating small expectation of profitability.



In the training environment, the rewards are also scaled; the reward is the percentage change in portfolio value from the previous day. Thus, it is centered at 0 and in [-1,1].

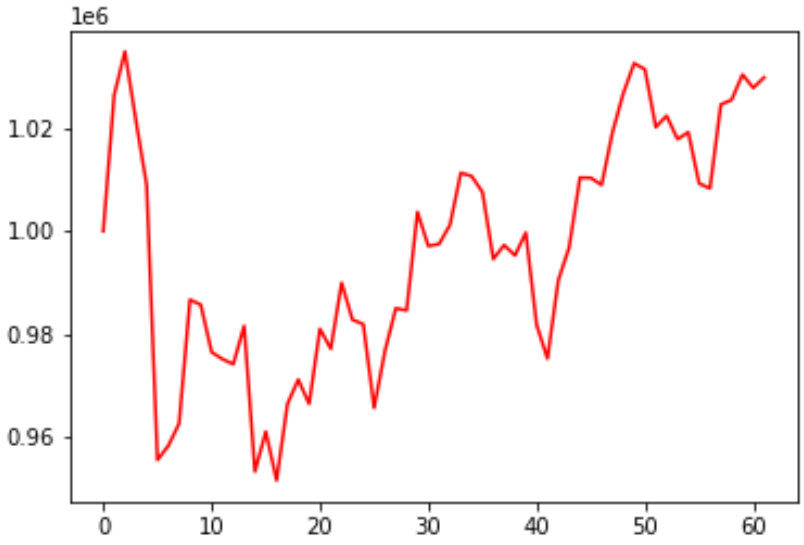
The rewards are not scaled in the validation or testing environments, as they are not being used to train the model.

The plot below shows system performance in the training period. It is quite good, and in fact better performance than the results shown in the previous section.



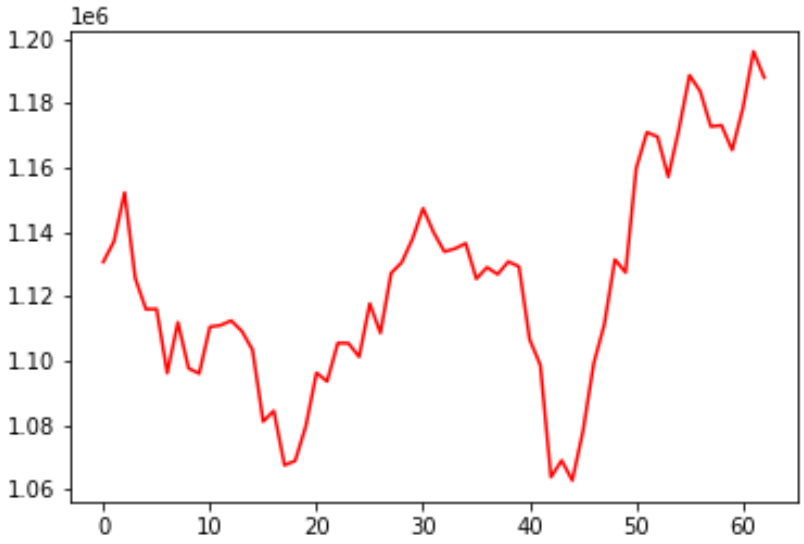
The algorithms fit the training model very well, though they do not avoid the precipitous fall in the beginning of 2020.

However, the system does not perform well on validation data. The plot below shows a typical validation window.



In validation, the system often meanders somewhat randomly between positive and negative growth; the reinforcement learners cannot accurately predict optimal actions. This is evidence of the algorithms overfitting to noise in the training data.

The following plot of a typical testing window shows performance similar to that in the validation window, further indicating overfitting on the training data.



Overfitting the training data often indicates that the predictive model is tractable; the model has enough information to learn to the desired function. Thus, this version of the system in which overfitting has a detrimental effect on overall performance still yields useful information. Based on these results, we have evidence that the methodology behind the system is valid. It will work, it just isn't working *yet.*

Conclusion

Significant work remains to get this trading system into deployable condition, but this preliminary report highlights significant evidence in favor of pursuing the ensemble reinforcement learning strategy. The fact that the model with scaled observation and reward spaces overfits the training data is especially encouraging; it indicates that the algorithms are finding actionable, reliable patterns. This also indicates that the state as used in the model should be sufficient for a properly fit trading system; no new technical indicators should be required.

The next step will be repeated experimentation, tuning, and performance analysis; tweaking the model until it fits properly and provides the out-of-sample performance of which it should be capable.

The model may suffer from the curse of dimensionality, trying to predict the number of contracts to buy in 78 different dimensions, and so limiting the contracts available or the maximum investment in each may provide significant benefit, from a machine learning and risk management standpoint.

The A2C, PPO, and DDPG algorithms were trained for a relatively small number of timesteps. They were trained for 100,000 timesteps each, while those same algorithms are often trained for millions of timesteps in other contexts, such as gameplay and robotics. However, the fact that the model overfit the training data and underfit validation could indicate that 100,000 is too many training timesteps, and less training might help the models fit better to new data.

The problem with training the models for many timesteps is that there is limited data available. We used only 3007 days of data, total. Training for many timesteps on a small amount of data could certainly lead to overfitting, but training for fewer timesteps on a small amount of data could certainly lead to underfitting the training set. Thus, it may be worthwhile to pre-train the algorithms on large sets of synthetic data. Generative Adversarial Networks (GANs) could be used to synthesize data with the same characteristics as the real data, and then the reinforcement learning models could be trained for millions of episodes as is often necessary. There is some debate over whether using synthetic data for pretraining is advisable; some say that it is just another form of overfitting, since we have no reason to believe that the true patterns in the real data will occur in the synthetic data, but synthetic data is regularly used for financial machine learning because real data can be quite limited, especially at larger time intervals.

Experimentation and improvement of the system will continue, and updates will be reported monthly until the model reaches the deployment stage.