MTH9899 Final Project Presentation

Yigao (Hugo) Liu, Haocheng (Frank) Gu,
Chenyu (Phillip) Zhao, Zichao (David) Wang

Baruch College, CUNY

May 22, 2019

Overview

Correlation Heatmap

	Date	sec_id	fut_ret	vol	X1	X2	X3	X4	X5	X6	X7
Date	1	-0.000360042	0.000959238	-0.037649	0.00178204	0.00150207	0.0154429	-0.0109058	-0.00469048	0.0328281	-0.00314545
sec_id	-0.000360042	1	0.00386634	0.294814	0.00979511	-0.00243256	-0.00658754	-0.0231365	0.00279693	0.0175532	0.000984823
fut_ret	0.000959238	0.00386634	1	0.0108265	0.0296427	-0.00100191	0.00238215	0.00126917	-0.00137804	0.00457898	0.00529302
vol	-0.037649	0.294814	0.0108265	1	0.0938776	0.00325418	-0.0521775	-0.0483511	0.00206698	0.0121984	0.021107
X1	0.00178204	0.00979511	0.0296427	0.0938776	1	0.00251701	-0.0372248	0.00555938	0.00154414	0.016531	0.0636026
X2	0.00150207	-0.00243256	-0.00100191	0.00325418	0.00251701	1	0.00211169	0.00138603	-0.00236831	0.000528103	-0.000807402
X3	0.0154429	-0.00658754	0.00238215	-0.0521775	-0.0372248	0.00211169	1	-0.0310279	-0.00177341	0.0581475	-0.0944369
X4	-0.0109058	-0.0231365	0.00126917	-0.0483511	0.00555938	0.00138603	-0.0310279	1	0.000967802	-0.0511142	0.0288947
X5	-0.00469048	0.00279693	-0.00137804	0.00206698	0.00154414	-0.00236831	-0.00177341	0.000967802	1	-7.36434e-05	-0.000131317
X6	0.0328281	0.0175532	0.00457898	0.0121984	0.016531	0.000528103	0.0581475	-0.0511142	-7.36434e-05	1	-0.0727202
X7	-0.00314545	0.000984823	0.00529302	0.021107	0.0636026	-0.000807402	-0.0944369	0.0288947	-0.000131317	-0.0727202	1

Figure: pairwise correlation heatmap

- "sec_id" has a strong positive correlation with volatility.

Population Histogram

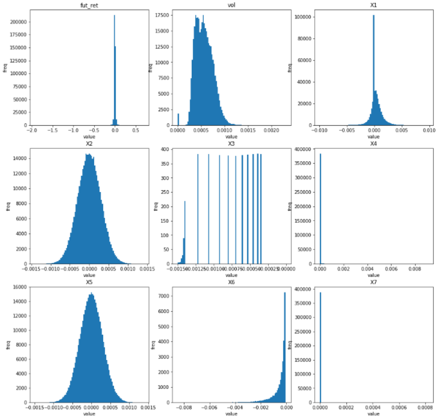

Figure: population histogram for each feature

- "X2" and "X5" are very likely to be pure white noise.
- "X1" seems to contain some information.

Moment Plot

One way to further investigate whether a feature "X*" is noise is that for each ticker, we plot the 1st, 2nd and 3rd moment of "X*" across the time, and see if the value varies across each ticker.

- If it does not vary, "X*" is probably noise.
- Otherwise "X*" may contain some information.

Moment Plot

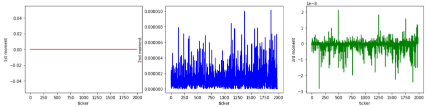

Figure: 1st, 2nd and 3rd moments of "X1" across different tickers

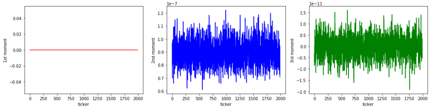

Figure: 1st, 2nd and 3rd moments of "X2" across different tickers

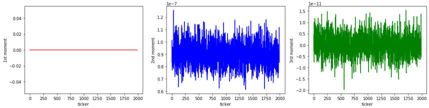

Figure: 1st, 2nd and 3rd moments of "X5" across different tickers

Two Possibilities

There are two most common possibilities for missing values in a time series:

1. A few np.nans lie among valid data in a time series.
2. A big chunk of np.nans at the beginning (or in the middle) of our time period.

We can probably use some kind of moving average to fill the np.nans in case 1; but we have to treat case 2 more seriously.

np.nans in "vol"

- One way to fill np.nan is to use exponential moving average.
- We calculate the moving average using historical data in order to avoid lookahead bias.
- But there is one parameter we have to decide: the center of mass (or effective lag) of our moving average.
- Therefore, we have to do some kind of parameter tuning.

Tune with "sec_id"

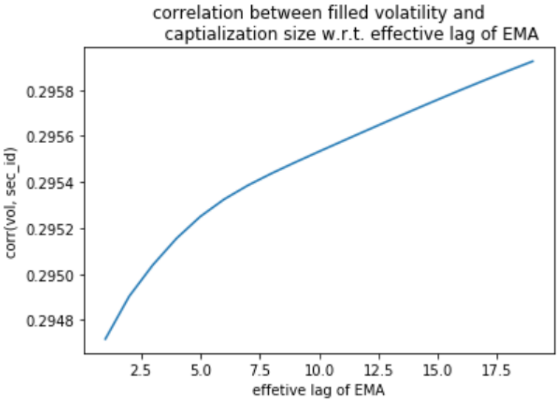

Figure: correlation with EMA-filled vol w.r.t. sec_id

- We are disappointed to see that the correlation keeps going up, and a EMA with $lag > 15$ actually does not make sense.

Tune with "X1"

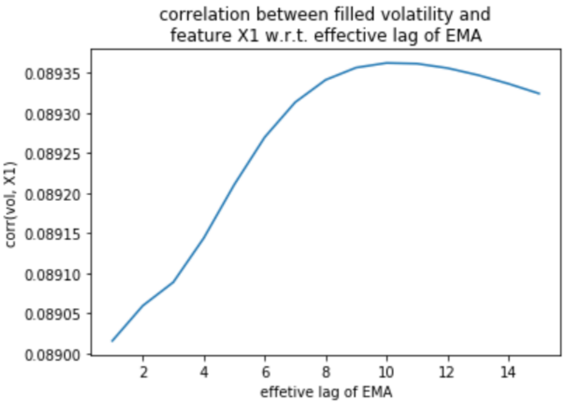

Figure: correlation with EMA-filled vol w.r.t. sec_id

- Based on this plot, we will choose $lag = 10$.

Remaining np.nans in "vol"

- After filling in with exponential moving averages, we can find that there are still 20,887 np.nans in "vol".

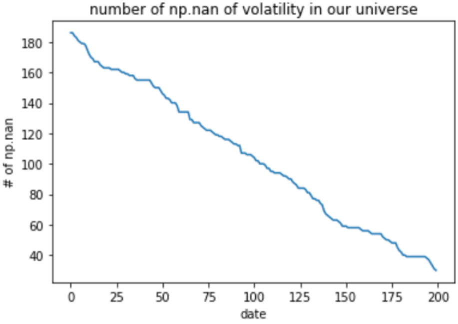

Figure: number of np.nan of volatility w.r.t. date

- The most reasonable way tends to be the most naive one: just leave all np.nans unfilled.

Normalization

We set our features into four sets and treat them differently:

1. sec_id, Date: we will divide each value by the sample maximum.
2. X2, X5: we will not normalize them because we will drop them in the end anyway.
3. vol: we will first add a minor positive number (10^{-4}) in order to avoid numerical error, and then take log and compute z-score. New feature will be named "log_vol".
4. X1, X3, X4, X6, X7: we will compute z-score of them directly. New features will be named "X*_norm".

Different Regressions

- Models: OLS, +/- 4MADs, OLS on the strongest signal, OLS by large and small SECID, LASSO, Ridge.
- Experiment: LightGBM.

R2 of Different Models								
Type	OLS	Use +/- 4MAD	OLS with Strong	L/S SEC ID	LASSO	OLS Drop Date SEC	Ridge	Light GBM
In Sample	0.084263%	0.096267%	0.082739%	0.081691%	0.083442%	0.082275%	0.084263%	0.311028%
Out Sample	0.115843%	0.111507%	0.108627%	0.119250%	0.117673%	0.117736%	0.115867%	0.092557%

Figure: R^2 of Different Models

Prediction vs. Actual

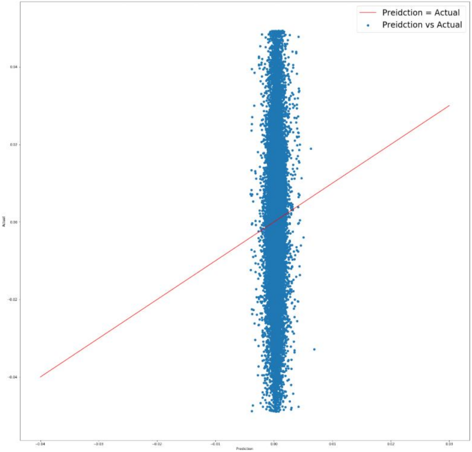

Figure: Prediction vs Actual

LASSO

- Date and SECID reach 0 first.
- X1 last till the end.

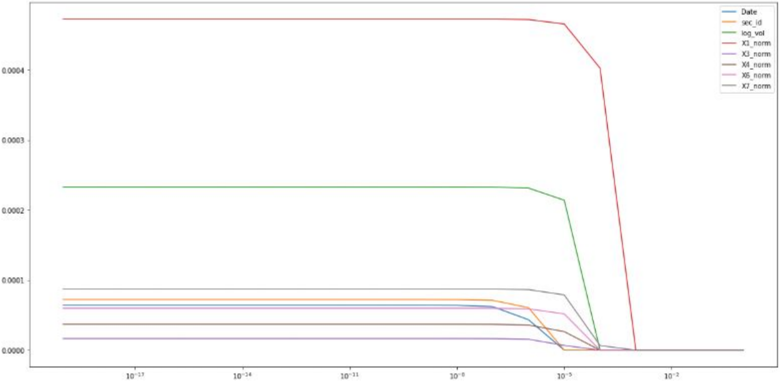

Figure: Coefficients of Variables as Penalty Increases

Tree Models

- Default tree
- Default random forest
- Self-defined tree
- Self-defined forest

Default Tree

- Randomly split
- Cross validation on depth
- Other parameters

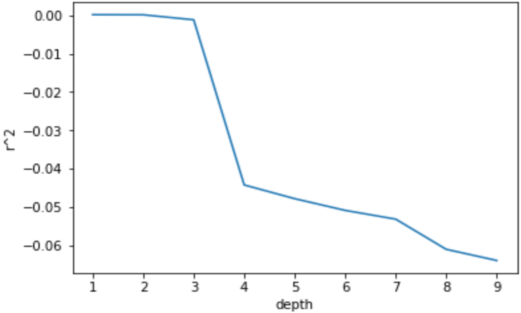

Figure: depth of default tree

Default Forest

- Use 3 as the best depth
- Cross validation on the numbers of trees
- Other parameters

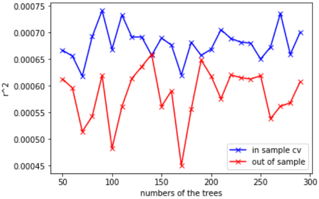

Figure: cv and r^2

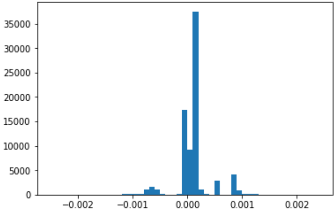

Figure: hist of output

We can see that the performance is not very good, and the predictions are just the mean.

Self-defined Tree

- For each feature, distinguish zero from non-zero values first
- Cross validation on the depth
- Other parameters

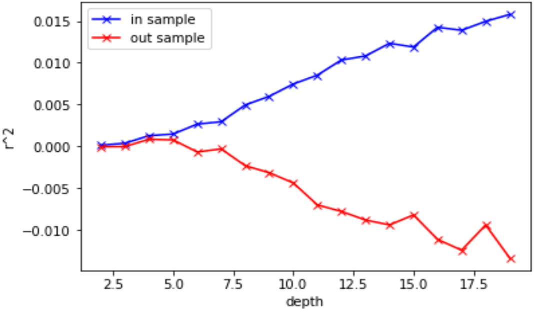

Figure: depth of self-defined tree

Self-defined Forest

- Use 7 as the best depth
- Cross validation on the numbers of trees
- Other parameters

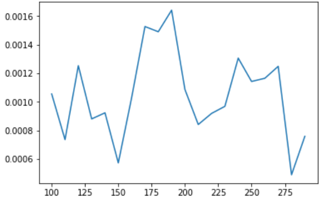

Figure: cv of our tree

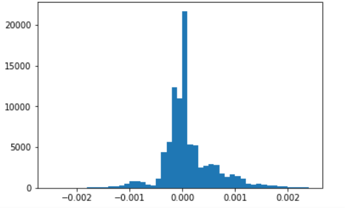

Figure: hist of output

We can see that the performance is better than before. Although there are still many predictions which give the mean value, we can see a fat tail in this picture. The average out-of-sample r^2 is 13 bps.

Adding More Randomness

- Randomly select 3 features to split on each point.

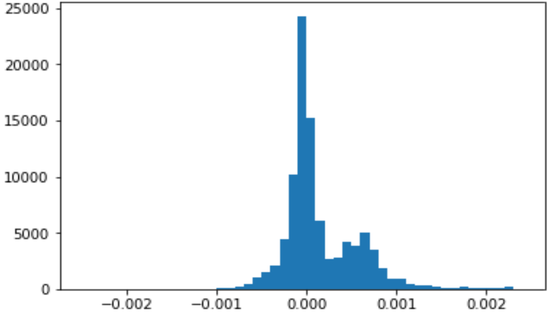

Figure: hist of output

We can see that the performance is still better than default.The shape of histogram is almost the same as before, but the average out-of-sample r^2 is 10 bps.

Neutral Network and its results

I applied different variations of Neural Networks to different variations of data set, turned out no good result benefited from deep net structure.

- NN can not capture fat-tail of return distribution, under scale of original data.
- There is no good metrics, NN behaves poor in both train and test set.
- Best test R-square is between 7 to 12 bps, when framework is very simple
- When build it deep, NN tends to predict the mean value

Histogram

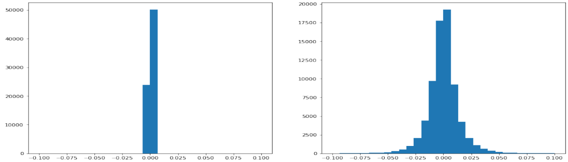

Figure: Best Prediction / True results

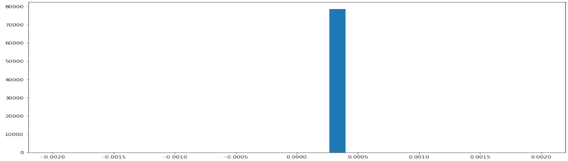

Figure: Deep NN's Prediction

What I tried

- Structure: linear fully connected NN, up to 5 layers, 128 neurons each
- Optimizer: Adam
- Different learning rates
- Activation function: ReLU, leaky ReLU, Tanh
- Drop layer: randomly inactivate 0.5 portion of neurons in each epoch
- Batch normalization layer: normalize output after each layer
- L2 regularization: on weights of neurons

Best network

- single layer, 4 neurons
- no drop layer
- L2 regularization: 0.1
- leaky ReLU
- do batch normalization
- use sec_id, X1_norm, X3
- to predict fut_ret/vol

Can basically predict rise or drop. R-square between 7 and 12 bps in both test and train sets.

Aggregation

- For aggregating, firstly we trained all existing models under first 150 days. Then these models gave predictions for latter 50 days.
- Now we want to use these predictions from existing models as input for an upper-level model. This upper-level model will combine all results from existing models.
- We split these 50 days into 3:1 portion as training and test set for this upper-level model.
- Note that the training set for this upper-level is the test set from lower-level. From this point, training or test is in sense of upper-level model.

How to deal with NaN in vol

We used two different ways to deal with NaN in vol.

- First way: We fill all NaN with 0.
- Second way: We treat data with NaN vol as another group of data. We assume it has different property from other data. Under this consumption we build our two-level model only on non-NaN-vol data, and build another simple model for NaN-vol data.
- For the latter simple model we use linear regression on all other features, instead of just fill 0 in return prediction. Because we don't want to assume any extra things. Linear model is most explainable, so we choose it.
- Actually we model two kinds of data separately and combine them together

These two methods has close performance up to lower-level. But when we do aggregation, we found the second one has much better r-square. Though the second way maybe have data-driven suspicion, latter we will focus on this method.

Existing model

	0	1	2	3	4	5
0	1	0.872693	0.90927	0.638748	0.672003	0.581991
1	0.872693	1	0.948424	0.705184	0.735551	0.671929
2	0.90927	0.948424	1	0.685353	0.720101	0.653424
3	0.638748	0.705184	0.685353	1	0.789573	0.712224
4	0.672003	0.735551	0.720101	0.789573	1	0.91553
5	0.581991	0.671929	0.653424	0.712224	0.91553	1

Figure: Correlations between features in train set

The existing models from lower-level are Neural Network, OLS1, OLS2, Ridge, Light GBM, Random Forest1, Random Forest2

Agg method

- Neutral Network: Even after the simplest net, the prediction is mean of data.
- OLS: Bad approach, as there is high correlation between features
- Lasso: very small penalty(1e-7) leads to choose only one feature. We let it choose two features.
- Ridge: Best results cannot beat Lasso.
- Random Forest: Very amazing!
- Light GBM: Not amazing.

As usual, all models' hyper parameter tuned with cross validation.

Results

	train	test
AGG_Forest	0.013336	0.002272
AGG_Las	0.001555	0.001332
AGG_Rid	0.001586	0.001304
Regression1	0.000995	0.001214
Forest1	0.001440	0.001207
Forest2	0.000953	0.001166
GBM	0.000342	0.001105
NN	0.001097	0.001088
Regression2	0.001059	0.000960
AGG_LGBM	0.003398	0.000712
AGG_OLS	0.001893	0.000672
AGG_NN	-0.000002	-0.000038

Figure: Results(in order of test R2)

Conclusion

So aggregating did give us better result. So we choose Random Forest aggregation with 6 other models as our final choice. feature importance:

- NN: 0.26931832
- Regression 1: 0.1825856
- Random Forest 1: 0.17319282
- Random Forest 2:0.14946527
- Regression 2: 0.13878568
- Light GBM: 0.08665231

Under this method we can achieve 23 bps while the first fill-NA-with-zero way can achieve 13 bps.

Results

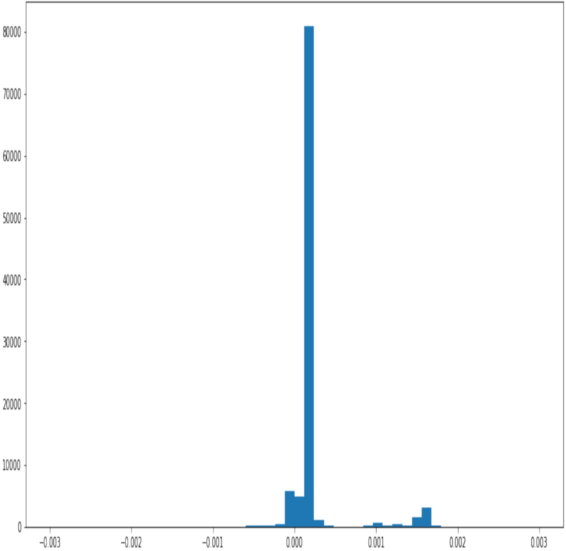

Figure: oos prediction histogram

Thanks for Listening