Practical tips for algorithmic trading

Evgeny “Jenia” Mozgunov, Caltech

Creating a winning trading algorithm

1. Get familiar with the **trading platform** and its limitations
2. Set up **optimization** and **crossvalidation** routines
3. Come up with several strategies and backtest them using 2.

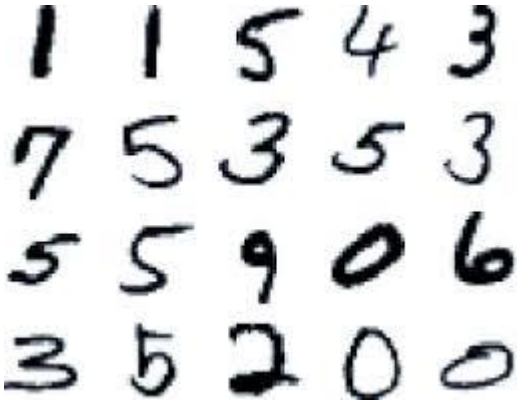
Every step has its own
tricks of trade that may be counterintuitive for the beginner.

Enter machine learning

Some economists think that stock prices fundamentally cannot be predicted.
To be confident in either point of view, one should take a machine learning approach to the problem.

Machine Learning vs. Finance

Classic machine learning task is image recognition.



Q: What is in the picture?

In contrast, price prediction can be thought of as:

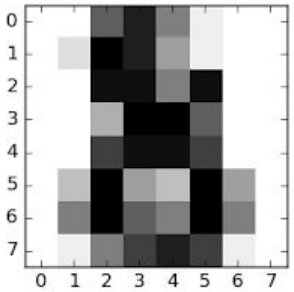
Q :What is in the picture? But I don't show you the picture, I hold it face down.

Indeed

We are trying to predict what's gonna happen tomorrow. The best way is to wait & see.

The processes that are market making tomorrow most likely have not even started today. And those are random processes.

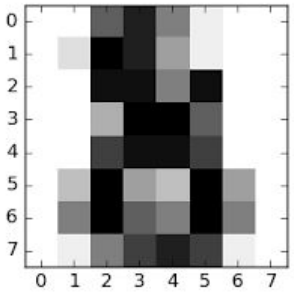
More specifically:



Difference between finance & Machine Learning:

- Noise!
- It's not enough to predict well, one should find out how to make money using a prediction.
- Danger: we are prone to gambling.

Noise definition

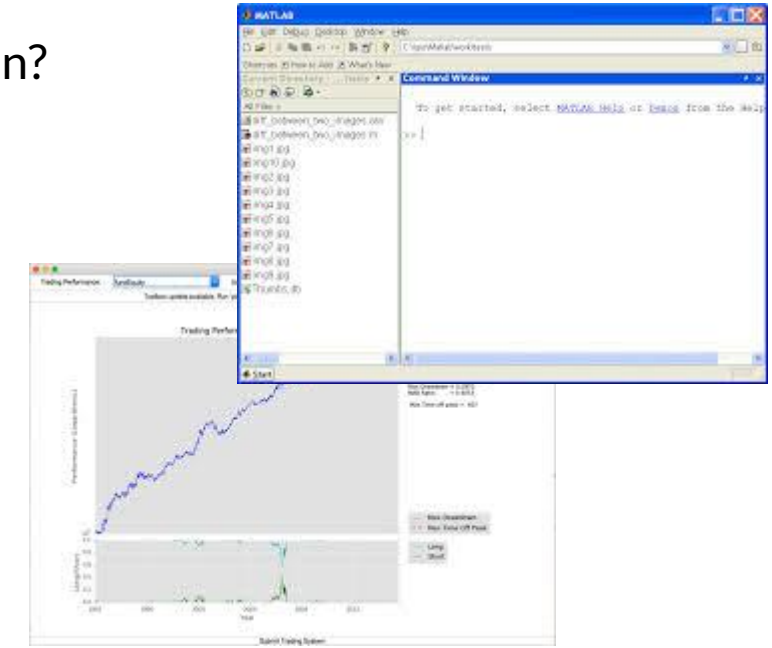


Noise - contributions to the data that are completely inaccessible to us and that cannot be modeled with information available to us.

In the case of the stock market: what other players are doing, what other players are thinking. Unpredictable events. Insider information.

So it's a hard problem

Ready to dive in?

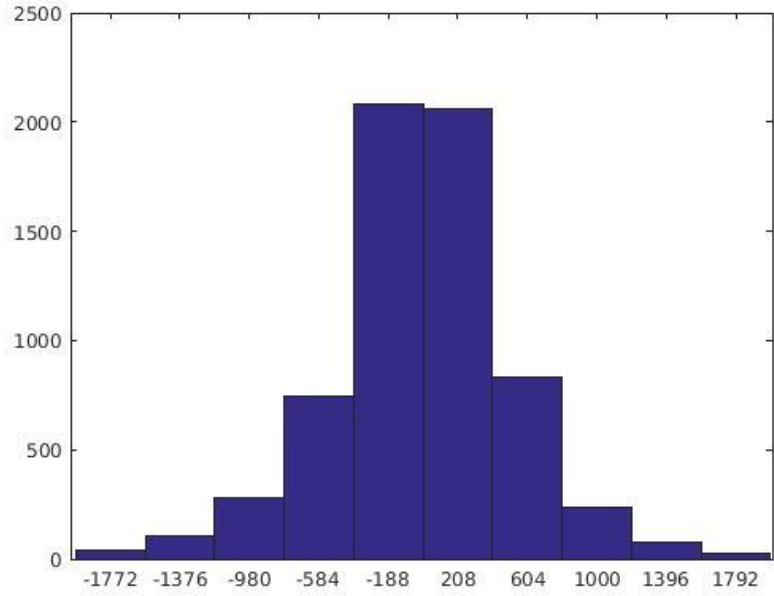


QUANTIACS

Let’s go!

Histogram for price change:

```
>>s = getSettings('newts');
data = loaddata(s);
data.CLOSE(-1,1);
dP = data.CLOSE(2:,2) - data.CLOSE(:(end-1),2);
dPnum = dP(~isnan(dP));
dPbound = dPnum(abs(dPnum)<2000);
>>hist(dPbound)
```



Our best prediction:

Nothing’s gonna change.

One can check that it gives the best Mean Square Deviation of predicted price from actual price.

But wait

We'd like to make money. If $\Delta P = 0$, we don't feel like buying or selling.

So the chosen objective function **Mean Sq Dev** was not appropriate. We need to choose an objective function that prefers predicting $\Delta P \neq 0$, even if at the cost of increasing **Mean Sq Dev**.

Is that possible?

How can we make money on average, if we are worse at predicting than the guy that says “Nothing’s gonna happen”?

Our Profits = $\sum$ Decision(prediction)*ΔP_{true} >0

MSDev = $\sum$(prediction - P_{true})^2

$MSDev_{trivial}$ = $\sum$(ΔP_{true})^2 MSDev > $MSDev_{trivial}$

Constructive proof

It is possible! Let

Decision(prediction) =sign(prediction)

Let ΔP_{true} be -2,-1,+1,+2 with probability ¼

prediction=-1,+2,-2,+1 for each of them

MSDev = 20>10 = $MSDev_{trivial}$

Expectation Value (Our Profits) =+0.5

Normalization

Here Decision = sign(prediction) = ±1

But if we set Decision =±10 (bought or shorted 10 times more)

We get Expectation Value (Our Profits) =+0.5*10 = +5

That's why we didn't just set Decision =prediction - it would lead to ∞ profits if we choose to big prediction.

Normalization

So “max ∑Profits” lead to ∞’s unless we bound Decision.

Another way to get rid of ∞’s is to normalize the objective:

$\max \sum Profits / \sqrt{\sum Profits^2}$

Which is just:

max Sharpe Ratio

Now we can make Decision arbitrary big.

First Conclusion

One needs to see which decision maximizes the expected value of the Sharpe Ratio, even if one cannot predict the prices better than trivial prediction.

Conveniently, in the Quantiacs competition, Sharpe Ratio is the criterion used to choose the winner.

Decision vector in Quantiacs

```
p = ones(1, nMarkets);
```

What is p?

|p(k)| is a fraction of our net worth that we decide to use to buy (or short if p(k)<0) k'th futures.

p(k) = Nshares(k)*price(k)/TotalMoney

ret(k) = Nshares(k)*Δprice(k)/TotalMoney

$\sum_k$ret(k) = ΔTotalMoney/TotalMoney

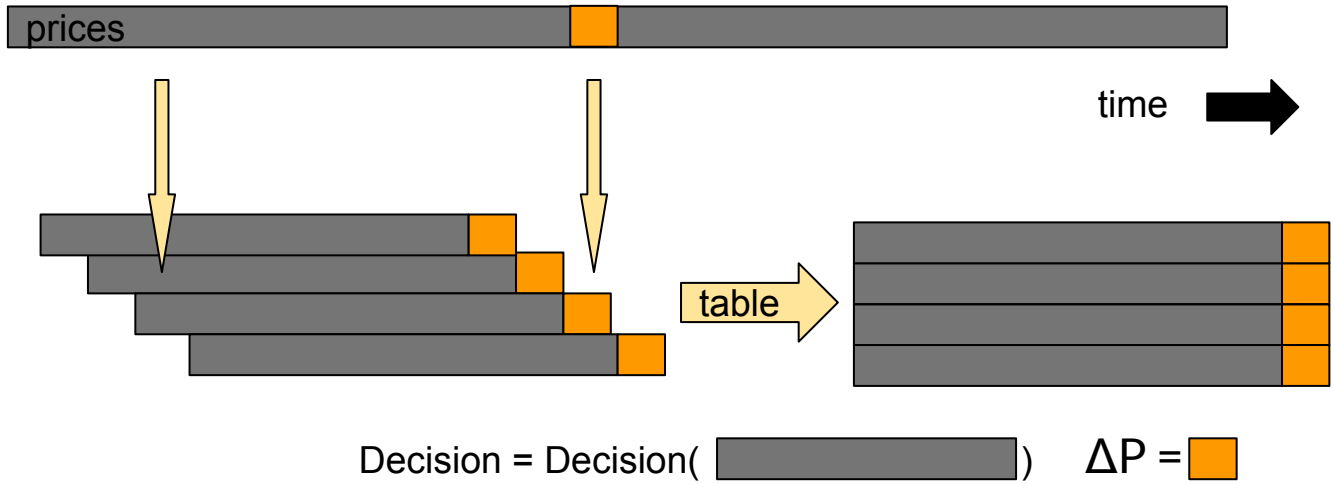
Check yourself

- It is automatically normalized
- See runts.m - TotalMoney is “fundEquity”
- Variable “Equity” has no meaning

In our notation p = Decision

How to make decision

Any time series can be broken down into features by duplication:



Examples

Toolbox provides MeanReversion and TrendFollowing:

Decision([redacted] , MR)
Decision([redacted] , TF)

In fact, each of the two algorithms has parameters:

Decision([redacted] , MR, periodLong, periodShort)
Decision([redacted] , TF, periodLong, periodShort)

Generalization

All of our algorithms can be written as:

Decision([redacted] ,parameters)

Parameters = WhichAlgorithm, periodLong, periodShort

Let’s not forget that we have multiple futures, so we can use different algorithm on each of them and combine. But how to do it?

Choosing shares of a portfolio

For each futures, $\text{Decision}_k = \pm 1$ or 0 for MR and TF

Then these algorithms split portfolio equally between the futures they decided to trade.

Suppose we came up with weights that represent our confidence in each futures:

Decision = Combine(Decision_k, Method)

Full list of parameters

We can add portfolio weights Method to our parameters :

Decision([redacted] ,parameters)

Parameters = $\{$WhichAlgorithm, periodLong, periodShort$\}_k$, Method

Note that now every futures can be traded with its own parameters.

Optimization

While our parameter space is not very big, we can just try everything and see which one gives the best SR on the backtest:

$$\max_{\text{parameters}} \text{SR}$$

I tried $\max_p$ SR

This is what I submitted to Q2:

I lost badly...

Generalization

All of our algorithms can be written as:

Decision( ,parameters, hyperparameters)

The distinction between parameters and hyperparameters is completely artificial!

Finally Machine Learning

One splits the data into training set and test set, and runs two optimization programs. On training set:

$\mathbf{max}_{\mathbf{parameters}}$ **SR** with hyperparameters fixed

Then using found best parameters for each value of hyperparameters, go to test set and do:

$\mathbf{max}_{\mathbf{hyperparameters}}$ **SR**

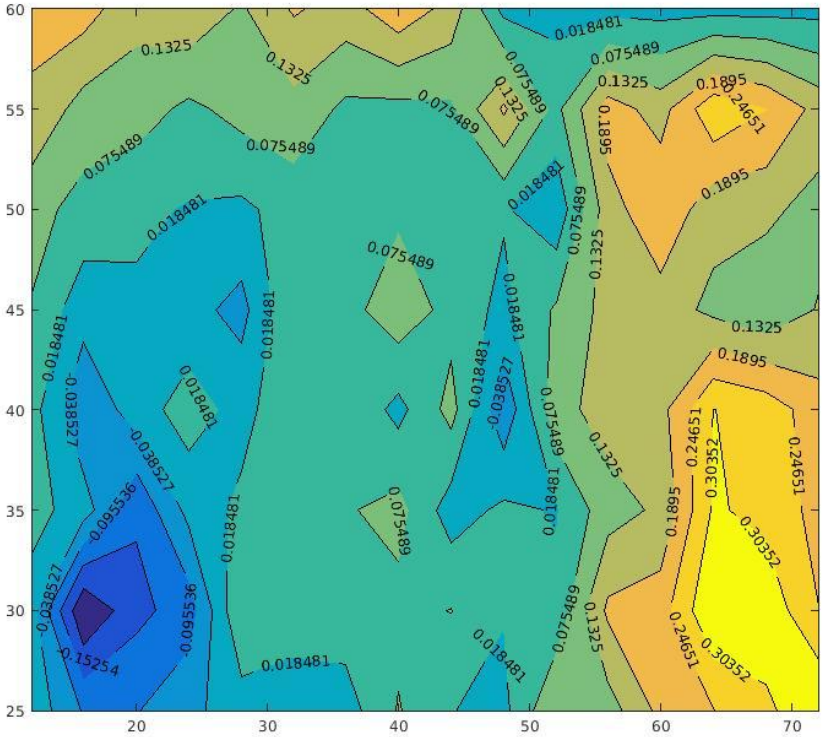
Reference

“Tuning hyperparameters”, Wikipedia

Example of hyperparameter Crossvalidation:

Blue - algorithm loses money

Yellow - algorithm wins money



Topics not covered:

- Commission and slippage

My answer to economists who think it is hard to predict stock prices: it’s actually very easy, but you can only predict movements that bring less profits than fees to do them.

- Overcoming greed
- Personal financial decisions
- Time to start a hedge fund?

Thank you!