Building AI Trading Systems

Denny Britz

2020-06-26

About two years ago I wrote a little piece about applying Reinforcement Learning to the markets. A few people asked me what became of it. So this post covers some high-level things I've learned. It's more of a rant than an organized post, really. If there is enough interest in this topic I'd be happy to go into more technical detail in future posts, but that's TBD. Please let me know in the comments or on Twitter.

Over the past few years I've built four and a half trading systems. The first one was crap. The second one I never finished because I realized early on that it could never work either. The third one was abandoned for personal and political reasons. The fourth one worked extremely well for 12-18 months, passively producing on the order of 1-2 full-time salaries with a relatively tiny investment. Then, profits started decreasing and I decided to move on to other things. I lacked the motivation to build yet another system. Some of the systems I worked on were for the financial markets, but the last one was applied to the crypto markets. So keep that in mind while reading.

Trading as multiagent problem

If you've taken some economics classes your first thought is perhaps something like

> Ha, Efficient Market Hypothesis! You can't beat the market! It's all just luck!

And I would say that most economic theories like the EMH are quite far removed from the real world. This doesn't necessarily mean they're wrong. They are interesting academic models that can be useful in certain contexts, like for writing papers or getting prizes. But they don't survive confrontations with a messy real world and emotional human beings. I won't get into it here because you can find smarter people arguing about this, but even the original author admits that the EMH is just a theoretical model. Suffice to say, I believe (and have seen myself) that you can build profitable systems that consistently beat the market. Will the market eventually adjust and take away that opportunity? Probably, but that's irrelevant, because the market may take months, years or decades to

adjust (just look at HFT), and you are free to evolve your system over time, looking for new opportunities and adjusting to the adjusting market yourself. That was the case for me. The market may have adjusted to my system as profits started decreasing, but it took more than 12 months, and with the right motivation I probably could have adjusted and improved the system to work again.

Rather than thinking about markets from an economics perspective, I prefer to think about them from a Game Theory or Reinforcement Learning perspective. **The market does not exist.** What people call the market is an emergent property of many agents acting in an environment trying to maximize their own objective. The objective may be slightly different for everyone. Some want to sell shares they obtained through a stock options plan at a good price. Some engage in arbitrage for quick profits. Some try to predict short-term movements based on news or charting patterns. Some engage long-term speculation based on fundamentals. Some are designated market makers providing liquidity for a specific asset or exchange in return for a fee. Each could be a human acting on gut feeling and emotion, a human following some kind of implicit fuzzy algorithm (e.g. charting), or an automated system acting on data. The combined behavior of all these agents gives rise to this illusive thing that people call *the market*. Your goal isn't to beat the market, it's to beat some of the other players in the market over a time horizon you care about. The time horizon part is important. Not every trade has a clear winner and loser, because agents are optimizing over different time horizons, ranging from microseconds to decades.

A trader's mindset: It's all relative

If you come from a tech or startup background, transitioning to trading may require a change in thinking. Products and engineering are often about **absolutes**. If your website takes 100ms to load that's pretty good. Making it load in 99ms provides negligible benefit to the end-user. It'd be a waste of engineering resources. The same is true for startups. Paul Graham likes to say that startups are rarely killed by competitors. Rather, they commit suicide. They fail because they cannot find customers, don't have a solid business model, or because of founder issues. Being killed by competition with a slightly better product is quite rare.

Trading is different. It's about **relatives**. It's fine if your website takes a horrible 10 seconds to load if your competitor needs 11 seconds. Having crappy data is fine if everyone else's data is even crappier. Paying high trading fees is fine if everyone is paying the same. This is pretty obvious if you look at the market as a multiplayer game. Even if you're a bad player on an absolute scale, you can win if your competition is worse. This has direct effects on how you build software to trade in the markets.

Building trading infrastructure

What a focus on relative performance means for engineering can be quite counterintuitive.

When building infrastructure for a product-based company you would prefer using battle-tested, robust frameworks that minimize risk. Why re-invent the wheel if someone else has already built it? It's safe to use something that has been adopted by thousands of people. This is where open source shines. Using AWS for hosting your server? Postgres for storing your data? JSON for serialization? Python or Node for data collection? An open source library for calling APIs? Download Keras and train a NN on your data? Reasonable ideas, but when you are building trading infrastructure, you should think twice about these.

Widely adopted technologies are commodities. If you use the same technologies as everyone else, where will your advantage come from? When building a product this doesn't matter because it's about absolute SLAs. But trading is about relatives. **Each commoditized technology you can specialize and build yourself is an opportunity** to beat the competition. By being hybrid cloud you can reduce latencies. By storing data in an efficient binary format you can save serialization time, leading to faster iteration during development, and faster predictions in production. By creating custom integrations with exchanges instead of using off-the-shelf APIs you may be able to use specific order types that are not available to others, or gain an informational advantage by processing exchange-specific information. And so on. Building highly-specialized non-generic systems is what engineers hate. We love generalization and abstraction. But specialization is often where your advantage in trading comes from.

You don't need to start from scratch and implement your own programming language to start trading. But you should be aware of the tradeoffs you are making when adopting commoditized technologies. Make conscious decisions about it. Some of the things above seem minor - but a large number of tiny advantages can add up to a significant edge.

AI and trading

You may say, well, this makes sense, but I really don't want to compete on the infrastructure side. Instead, I want to build a smarter model that makes better predictions. That's my edge! I've heard this a LOT, and I've never seen it work.

AI is pretty commoditized. I would say it's more commoditized than excellent infrastructure engineering, data collection, or inductive biases from domain knowledge. These days, you can easily download state-of-the-art models and run them on your data. Unless you are at the forefront of some extremely relevant research, it's unlikely that you can gain a significant edge just from training a better model.

When people realize that their fancy AI model built on crappy infrastructure

trained with crappy data (that's also used by everyone else) doesn't work, they give up. AI can give you an edge, but not an edge so huge that it allows you to ignore other factors. You still need to build good infrastructure, get good data, have decent latencies, and so on. Few people seem to be willing to spend time on these unsexy things. Think of each as a multiplier. If one is close to zero, it doesn't matter how good your AI model is.

So, what are some these other things?

- **Market** - Pick the right market(s) to trade in. Don't go for the obvious choices that everyone is picking by default. The harder it is to get access to a market, both legally and technically, the more likely that you will find opportunities. Less liquid markets may be overlooked by sophisticated funds because they don't scale to their AUM. Again, this is often counter-intuitive to engineers who go for "good APIs" - good APIs typically mean popular. Popular typically means commoditized.
- **Data** - Think about data sources that others do not have access to, or are not willing to get. For example, there may be data that's difficult to crawl due to sophisticated rate restrictions IP bans. Most people will give up here. This is an opportunity for you. Be skeptical of popular APIs and open source software. It's the same data that everyone else uses.
- **Latencies** - You're probably not planning to compete with HFT traders (bad idea), but that doesn't mean you can completely ignore latencies. Better latencies will probably lead to easier execution with less slippage. Be careful about where you host your systems, how you send data around, how you serialize data, and so on.
- **Models** - In general, better data is more important than better models, but better models can also give you an edge. Note that you are often trading off model complexity with latencies.
- **Execution** - A good model doesn't mean much if you can't execute. Historical data you collect may look quite different from what's actually happening on the exchange. You are not going to know until you start live trading.

Reinforcement Learning

More than two years after my last post, do I still believe that Reinforcement Learning is the right approach to trading in the markets? Yes, and I've personally had success with it. However, I admit that a lot of tricks were necessary to make it work. The main benefit of Reinforcement Learning is that you don't need to set up a differentiable loss function. Instead, you can directly optimize Profit and Loss over some time horizon. By building a good simulation, your model can learn to be robust to latencies, jitter, slippages, and other things that may happen in a real market.

With supervised learning, it's a lot harder. There are many hyperparameters

you would need to get "just right" - What time horizon to optimize over? What to optimize (log returns? For what position size?)? When to exit a position? How to deal with stochasticity arising from latencies, rejected orders, API issues, etc? How to deal with non-iid data? And so on. I'm sure you can make that work if you try really hard, but relying on simulation-based approached just seems like a more principled solution.

I also believe that market simulations are a pretty good testbed for Reinforcement Learning algorithms. They have a few properties that many of today's techniques struggle with:

- Sparse positive feedback. With random exploration and naive rewards, you are quite unlikely to discover good policies.
- Requires generalization to future dates. Generalization is often overlooked in RL where researchers "test on the training set"
- With a good simulator, environments can provide a lot of stochasticity in the form of from latencies, jitter, API issues, slippage, etc.
- Nonstationarity. The market data distribution changes over time, and the agent must learn to deal with it.
- A low signal to noise ratio in the observations.

If you're an ML researcher interested in trading from an academic side, feel free to send me a message.

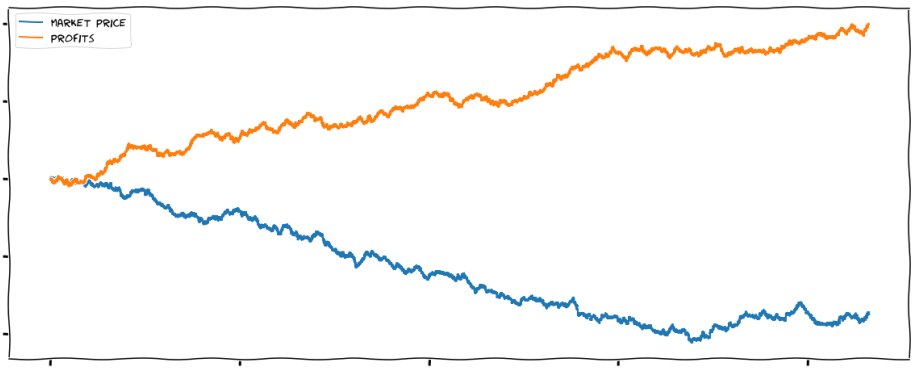
Optimize for Iteration Speed

Most inefficiencies are fleeting as other agents will adapt to your strategies. How fleeting? It depends. Depending on where and how you are trading, they could last milliseconds, seconds, minutes, hours, days, weeks, or months. **The way to be consistently profitable is to become consistently good at identifying fleeting opportunities.** This is basically meta-learning.

Optimize the speed of your outer loop when training models. The faster you can iterate on discovering new strategies and adapting your models and infrastructure, the better. Build your infrastructure around this. Querying and loading bulk historical data to play around with should be as fast as possible. Pay special attention to deserialization costs. When building a market simulator with matching engine, benchmark it extensively. Minimize RPC and network round trips. We're talking about orders of magnitude here. My first simulator was about 50x slower than my most recent one. Since the model is trained in simulation, that means 50x faster iteration speed when training or playing around. Automate data visualization so that you can look at them easily when you get unexpected results.

You can't make money when the market goes down

A common argument and misconception I hear from people who don't know much about trading is that I was lucky and made profits because the market was going up. Actually, many months my PnL graph looked something like this: (this is generated to get a point across, but my real data looked extremely similar):



So, did I just short the asset? No, I didn't short anything because shorting was not possible in the market I was trading in. **There is rarely such thing as the market going down all the time**. When people say the market is down, they talk about a specific time scale. The market may be down plotted on a daily or hourly scale. That doesn't mean there is only downward movement on shorter time scales. It may swing considerably. Once you start zooming into seconds and milliseconds, there will always both up- and down that you don't see on hourly, daily, or monthly charts. It's still possible to profit from that, without shorting anything.

Personally, I always made more during downward trends (on hourly or daily scales) than upward trends. I still don't know exactly why this happened. It could've been because that side of the market had more liquidity, or because more uninformed traders submitting naive orders were entering the market when prices started going down.

Learning about trading

Ask the same question to a dozen traders from different backgrounds and you'll get a dozen different opinions. Note that I use the word opinions, not answers or facts. Because trading is such a secretive field, there are very few widely accepted truths. People don't share the same background knowledge, and they use different terminology all over the place. If you want to learn about trading, it's hard. I don't know any online resources that do a good job at teaching

algorithmic trading. Most courses and tutorial you find online are written by gurus - they make money by teaching, but have never built a profitable system themselves. They'll probably try to sell you their SaaS product while they're at it.

The same is true for research papers. I subscribe to arXiv's q-fin.TR, but the average paper quality is quite low compared to fields such as Machine Learning. There are occasional gems with interesting ideas, but the vast majority are experiments by people trying to get job and to put something on their resume - none of it would hold up in the real world. Obviously, someone running a profitable system has zero incentive to publish a paper about it. Books such as Trading and Exchanges: Market Microstructure for Practitioners and Advances in Financial Machine Learning are a pretty decent starting point, but in my experience, nothing beats learning by doing or finding a mentor.

I may also write more technical articles about this topic in the future if there is enough interest.