mattturck.com /the-power-of-data-network-effects/

The Power of Data Network Effects

13-17 minutes

In the furiously competitive world of tech startups, where good entrepreneurs tend to think of comparable ideas around the same time and "hot spaces" get crowded quickly with well-funded hopefuls, competitive moats matter more than ever. Ideally, as your startup scales, you want to not only be able to defend yourself against competitors, but actually find it increasingly easier to break away from them, making your business more and more unassailable and leading to a "winner take all" dynamic. This sounds simple enough, but in reality many growing startups, including some well-known ones, experience exactly the reverse (higher customer acquisition costs resulting from increased competition, core technology that gets replicated and improved upon by competitors that started later and learned from your early mistakes, etc.).

While there are various types of competitive moats, such as a powerful brand (Apple) or economies of scale (Oracle), *network effects* are particularly effective at creating this winner takes all dynamic, and have been associated with some of the biggest success stories in the history of the Internet industry.

Network effects come in different flavors, and today I want to talk about a specific type that has been very much at the core of my personal investment thesis as a VC, resulting from my profound interest in the world of data and machine learning: *data network effects*.

Network Effects vs. *Data* Network Effects

The concept of network effect (in general) is by now well understood: a flywheel type situation where a good or service becomes more valuable when more people use it. Many examples out there from the telephone system (the value of a phone increases if everyone has a phone) to Facebook to many marketplaces (with some nuances for the latter).

While they produce many of the same benefits, *data* network effects are more subtle and generally less well understood. *Data network effects occur when your product, generally powered by machine learning, becomes smarter as it gets more data from your users*. In other words: the more users use your product, the more data they contribute; the more data they contribute, the smarter your product becomes (which can mean anything from core performance improvements to predictions, recommendations, personalization, etc.); the smarter your product is, the better it serves your users and

the more likely they are to come back often and contribute more data – and so on and so forth. Over time, your business becomes deeply and increasingly entrenched, as nobody can serve users as well.

Data network effects require at least some level of automated productization of the learning. Of course, most well-run businesses "learn" in some way from data, but that's typically done through analytics, with human analysts doing a lot of the work, and a separate process to build insights into the product or service. The more automation you build into the loop, the more likely you are to get a flywheel effect going.

Google is a classic example of data network effect at play: the more people search, the more data they provide, enabling Google to constantly refine and improve its core performance, as well as personalize the user experience. Waze, now a Google company, is another great example, essentially a contributory database built on data network effects.

There are also plenty of examples of data network effects found at the feature (rather than core business) level: for example, recommendation engines that are now everywhere from Amazon (products you'll want to buy) to Netflix (movies you'll want to watch) to LinkedIn (people you'll want to connect with), and keep getting better with more users/data.

Note that "standard" network effects and data network effects can absolutely be found in the same company. Part of the magic of Uber is that it benefits from both for its core mission: a standard network effect (Uber becomes more valuable for everyone as more drivers and more customers "join" the service) and a data network effect (more data enables Uber to constantly improve its routing algorithms to get customers a car as quickly as possible and to ensure its drivers get as many jobs as they can handle, making everyone happy and more likely to be long term members of the network). Similarly, Facebook benefits from both a "standard" network effect (the more people are on Facebook, the more interesting everyone's experience is) and data network effects, as the newsfeed, for example, keeps getting more personalized based on massive data learning loops.

Now available to everyone?

A lot of examples mentioned so far are about large companies, and indeed as we'll see below those large companies have a key advantage (more data).

However, data network effects are now becoming a possibility for a much broader group of companies, earlier in their development, as a result of the democratization of Big Data tools (cheaper, faster infrastructure to process large amounts of data) and machine learning/AI (an increasing number of off-the-shelf tools and algorithms to automatically analyze and learn from those large amount of data). In a world where you can have access to Google-like and Google-inspired technologies, from Hadoop to CockroachDB to TensorFlow, you don't need to be Google to put in place the core infrastructure and learning loops to benefit from data network effects. [disclosure: FirstMark is an investor in Cockroach Labs]

Data network effects can work equally well in a consumer context (learning across all users of the product) and an enterprise context (learning across all customers, that form a de facto network), across a variety of industries.

To take examples from different industries within my own portfolio:

- Productivity: As more users use Amy/Andrew, the AI-powered scheduling assistants created by x.ai, the system gathers more and more email data, which makes the AI smarter, which in turns improves the user experience (in terms of response time, for example) and makes the system ever more scalable (so it can serve more users), resulting in more usage and more data;
- Enterprise software: HyperScience (AI for the enterprise) and ActionIQ (Big Data Marketing platform) have both built systems that do (or will) get smarter with each new enterprise customer (both companies are largely in stealth);
- Internet of Things: Data network effects are particularly relevant to the long term defensibility of Internet of Things companies, including Helium (enterprise IoT) and Kinsa (health/consumer IoT) in my portfolio. Hardware will often get copied and sometimes will get commoditized. However, if you think of each device as a node in a network that contributes data, IoT companies have an opportunity to build insights/learnings from each customer that will be increasingly harder to replicate – the real value gets built at the software and data level;
- Health: Recombine, a genetic testing company, has built a network of partner clinics that administer its tests; with each new test, Recombine gathers more DNA data which (with appropriate consent) it can run machine learning on to improve its tests and nimbly develop new ones (therefore gathering more data);

- API/developer businesses: Sense360 is building an API to enable mobile developers to easily integrate sensor intelligence into their apps and will keep learning from massive amounts of data (GPS, accelerometer, gyroscope, barometer, etc.), across its network of customers.

The cold start problem and the "data trap"

Data network effects typically don't "just happen". To start with, they require a commitment from the startup to be fundamentally a data company, with a stated goal to build data feedback loops into the product, first manually, then automatically. This involves building the right type of data infrastructure (using modern Big Data platforms and tools) and data team (data engineers, data scientists, etc).

The other key requirement to build a data network effect is… well, data. Sometimes, it takes a lot of data. While you can do a bunch of things with small amounts of data, some of the more powerful machine learning algorithms (such as deep learning) are particularly data-hungry.

There's an interesting "chicken and egg" issue there — do you start by focusing on accumulating as much data as possible, and then build the data team/infrastructure, or the other way round?

Companies that aim to build data network effects at the feature level (e.g, commerce company building a recommender system to personalize their customers' experience) can afford doing the former and building their data team/infrastructure over time – see examples of recent talks by Birchox or Bonobos (where the data science and engineering team was created in 2012, 5 years after the company was founded in 2007) at our monthly event Data Driven NYC.

For data/machine learning "pure play" startups, building the data team naturally comes first, starting with the founders themselves, and the lack of access to large datasets at the beginning is a real issue. In part, it gives large Internet companies a real leg up ("why wouldn't Google do this?" becomes an even tougher question when it's not just about number of engineers but also about access to massive datasets). More fundamentally, having no or limited data, by definition, largely precludes significant progress on building a data-driven product — a real "cold start" problem.

The cold start issue can be more or less severe. In some cases, it can be surmounted reasonably quickly because the domain that the startup focuses on is comparatively narrow. x.ai is a perfect example of this: while the product keeps getting better with more data, the company was able to start automating chunks of the scheduling process remarkably quickly in large part because it addresses a finite universe of issues (there are only so many different scenarios involved in scheduling a meeting) and requires a specific type of data (emails that are relevant to scheduling a meeting).

In most cases, you need to have a data acquisition strategy to get over the cold start problem. I've seen too many data startups go to customers with an approach that often amounts to "give us your data, we'll use it to fine tune our algorithms, and awesome things will happen". Without real value provided upfront, that typically doesn't work.

A lot of the early stage data acquisition strategies I have seen succeed largely fall under the "hustle" category – a classic example of founders doing things that won't scale, in order to get the company to a stage where it may have a chance to scale. Some examples of strategies:

- crawling the web (to train entity recognition algorithms, for example)
- finding datasets on the Dark Web (legally, apparently)
- putting data capture SDKs in third party apps (with end user consent and for a fee to the app)
- doing some manual work to simulate what the software will eventually do
- making tiny acquisitions of companies to get access to a particularly relevant dataset

An approach I particularly like is building a "data trap". The idea is to build something that delivers real, tangible value to users from the beginning, and incite them to start contributing their data. For example, I have seen developer/enterprise startups building fun (and free) side apps, typically targeting consumers, to start gathering data. Sometimes, the "data trap" can be a real business unto itself — Recombine, the genetic testing company I mentioned earlier generates 8 figure annual revenues selling its pre-fertility test, a startling amount of revenues considering the data it gathers through those tests is highly valuable unto itself.

Learning from someone else's data

What if you don't own the data? This is a problem that applies across both consumer and enterprise businesses, regulated or not. Data privacy issues deserve an entirely different post, but the best practice here is to build privacy into the core DNA of the product from the onset. Disclosure, consent, user controls are essential.

In an enterprise context, the problem appears early and often. A typical scenario: a small enterprise software startup with great technology approaches a large corporation, promises to process and analyze large amounts of customer data, and by the same token hopes to fine tune its algorithms for the benefit of this customer (but, eventually, all other customers – to build the data network effect). The large corporation is very protective of its data, everything needs to be done on premise (and not in the cloud), and the security department will block anything it does not understand or cannot control. This can be a tough conversation.

Some possible ways of addressing this issue:

- Negotiating upfront and in full disclosure that, while the data will strictly remain the property of the customer, the data *learnings* will be owned by the vendor
- A contributory model where the customer needs to join the "customer learning network" to benefit from what the product learned from all other customers
- A tiered pricing where the customer pays more if they decide to not join the "customer learning network"
- Early in the life of the startup, targeting startups as prospective customers, as they tend to have a more progressive attitude towards those questions.

Conclusion

Nothing is ever easy, and getting data network effects going takes time, effort and often a lot of data. But the potential is exciting. The conclusion you should derive from reading this post is probably not that you should force fit a "data network effect" slide in your next VC pitch (!), unless it truly makes sense to have it there. However, I believe this is a very interesting way of thinking about how to build your business, particularly crucial for pure play data/machine learning startups but also relevant to a broad group of Internet and technology startups that are fundamentally data companies, and worth exploring early in the life of a venture, while making core infrastructure and team decisions.