Interview with Simon Funk

Gregory Piatetsky
KDnuggets
editor at kdnuggets

ABSTRACT

Interview with Simon Funk -- a Netflix prize leader, an outstanding hacker, and an original thinker.

Keywords

Netflix prize, SVD, collaborative filtering, artificial intelligence.

1. Who is Simon Funk?

Simon Funk became very well known in the recent Netflix prize competition (www.netflixprize.com/). At one point he was #3 on the list, and as of April 23, 2007 he shared 10-11th place with 6.31% improvement over the best Netflix result before the challenge.

Very impressive for an independent software developer who works on Netflix prize in his spare time between his trips around New Zealand!

However, Simon became well-known and respected not because of his result but mainly because he freely published his code and ideas – the first top leader to do so [1]. Hundreds of participants in the Netflix prize have used his code and explanations and have learned about machine learning and data mining in the process.

As I read entries in Simon blog (http://sifter.org/~simon/), I saw not only an accomplished developer and researcher, but also a very interesting person. Simon has shown his talents early, having obtained a BA from UCSD at the age of 18. He later was a lead developer for several outstanding systems, including

• Inkwell - a character recognition system used in Apple Newton 2.0 and re-released in MacOS 10.2 as "Inkwell"

• Adobe Atmosphere - a user-buildable, distributed, 3d virtual environment on the internet supporting avatars, chat, and linking between worlds.

For the last few years Simon lived in New Zealand. Before that he traveled widely, living also in California (in many places), Boston, Seattle, Portland OR, Maui, and Sweden.

I also found that Simon Funk is a pen name, and Simon's real name and bio are here (http://sifter.org/~brandyn/resume6.html). However, I will stick with Simon Funk for the rest of the interview, since this is the name people are familiar with in the context of Netflix prize.

Gregory Piatetsky-Shapiro: 1) What first attracted you to data mining / machine learning and what education did you receive?

Simon Funk: Laziness! Seriously, laziness is behind many of my projects and machine learning is the ultimate expression of this. (Never mind how many years of hard work it may take to automate a ten minute task...) I recall early on in my undergrad career trying to learn Lisp with a friend of mine and rapidly losing interest when we saw how much typing was involved. And not just the parentheses but everything between them too. I think about two hours in, I turned to him and said "I bet there's a tiny little program that would fit on a page and would learn all this stuff on its own."

Simon Funk

This seemed as good an excuse to quit as any, so we went out for pizza and forgot about Lisp. Not long after that I coincidentally met a guy who happened to be doing his thesis on neural networks (something I knew nothing about) and we got to chatting and he was telling me about this one page of code that learns all this stuff on its own... And of course my brain is going "Aha! I knew it!" which no doubt created deep Pavlovian associations forever endearing to me neural networks and their ilk.

Within a couple of weeks I'd trained up a net to recognize musical instruments by sound, thus solving the dreaded question of what I was going to do as a project for my computer music class...

And so a great partnership was formed: I write a few lines of code; the computer does all the work. I eventually received a B.A. in computer science from UCSD, with minors in math and psychology. In retrospect this was a perfect trio for machine learning, though in truth I picked them because they maximized overlap and minimized blue-book exams...

Part of that overlap was a psychology class taught by V.S. Ramachandran on vision and optical illusions, which was a great intro to the idea of perception as a mechanism. I also finagled my way into Robert Hecht-Nielsen's graduate neural network seminar where I got quite a broad exposure to the field over the course of the year. (Though largely I just remember sitting back and listening to him tell stories, and marveling at his keen taste in pocket kerchiefs.)

At the time, those guys didn't have the notoriety they do now, but they were both genuinely enthusiastic about the material, and it was interesting stuff, so those are the classes I remember. Who knows--maybe if my hydrodynamics prof had been as enthusiastic I would have become a plumber.

A few other inspirational tidbits stand out: I recall one talk I sat in on about using backprop to map from English letters to the phonemes of spoken English. The really interesting thing about this was that after the network was trained, they were able to create a sort of clustered hierarchy of categories for the letters of the alphabet, which is something linguists had been working on by hand for a long time. The results that popped out of this network in a day matched the linguists' graphs to a tee, but kept going where the linguists had left off!

I'm bushing over the details which I probably have wrong anyway, but I just remember getting this strong sense that wow: here was a class of tools that not only could learn things on their own, but can teach us. The way my Netflix algorithm discovers and reveals its own movie categories is a more recent example of the same thing, and I find it just as fun and inspiring now as then!

Another pivotal memory is, some time after college, reading Lynch and Granger's olfactory model. They go into some detail about how the olfactory cortex learns a hierarchical breakdown of smells through a cyclic process of successive refinement using reverse inhibition. Aside from just the interest in seeing how this sense of ours is actually implemented as an algorithm, it turns out that due to the timing characteristics of the various groups of neurons involved, the process works best when the stages of refinement cycle through at about four hertz--which is why we go "sniff sniff sniff..."!

So, here's this overt behavior we do, and the reasons for it trace back to the peculiarities of the particular algorithm our brain has evolved for learning and categorizing smells. If we can get inside of that, what about getting inside how we make our most basic choices, deciphering the mechanism behind the will itself? In fact this is largely what Granger is now pursuing at Dartmouth, so watch for some willful robots scurrying about their halls soon. (I may be headed there this summer myself.)

GPS: 2) Tell our readers about how you got involved in the Netflix Prize.

Simon Funk: Right around the time I was climbing on a plane for my trek around New Zealand (i.e., in some sense the worst possible time!), I got this enthusiastic email from my pal Vincent:

It looks like someone designed the contest of your dreams.

All you need to develop is the guts of the thing--no UI, no communications stuff, no infrastructure, no ancillary programs, no integration, no hardware or hardware interfaces. No need to recruit, pay, or work with a team. No tedious grant writing. No fund raising. No PR.

It looks like a pure pattern recognition problem that may be squarely in the path of where you want to go. Based on what I've seen, I think you could do this using some of the work you've already done.

There was more, but you get the idea. Honestly, I looked over the project and decided the odds of us actually winning were tiny (I still believe that), but since collaborative filtering is a big component of some of the projects I have in development, this seemed like a great data set to hone those ideas on.

GPS: 3) What led you to choose SVD approach over other methods. You have explained it well in your blog post [2], but can you briefly summarize the SVD approach?

Simon Funk: The best way to understand SVD is probably in reverse: to look at how one re-constructs a data matrix from the singular vectors. Consider just a single column-vector A and corresponding row-vector B. If you multiply A by B you get a matrix as tall as A and as wide as B. Now if you have some target data matrix of the same size (say the Netflix movie-by-user ratings matrix) you can ask: What choice of A and B would make that reconstructed matrix as close to my target matrix as possible?

SVD is a mathematical trick for finding that optimal AB pair. It's really just that simple, and the only additional tidbit is that once you've found that first AB pair, you can repeat the process with the leftovers (the difference between the original target matrix and AB) to get a second pair of vectors, C and D, and so on, such that the target matrix T is approximated by: $T = AB + CD + ...$, with each successive term being progressively less significant due to the "biggest bite first" nature of the basic algorithm.

Looked at that way, it's clear that SVD is a particular way of *modeling* the data matrix, T. Assuming we trust the math to find the optimal parameters (A, B, ...), the question is how well does the model reflect the true process behind the data matrix? Here A is a vector over movies, and B over users, and the matrix AB has in each cell the movie value from A times the user value from B. So in effect we are saying there is some aspect or attribute which is measured for each movie by A, such as "how much Action does this movie have?", and for each user by B, such as "how much does this user like Action movies?", and we just multiply those together to get our AB estimate for how much each user would like each movie based only on the amount of Action. The model further says that the contributions from the different attributes are just added up linearly to make a final prediction. And that's it.

It's a very simple model, but it feels qualitatively correct to a first degree of approximation. And I had a very simple way to implement it using the incremental SVD method that I had come up with previously in my attempts to filter spam among other things, so I gave it a go. Honestly I only expected it to do about as well as Netflix's posted baseline--I was quite surprised when it did as well as it did!

GPS: 4) What is your computing environment - hardware, software, and how much time you spend on development?

Simon Funk: For a few years now I've been doing pretty much everything on an ultra-portable laptop running Linux (currently Ubuntu on a PortableOne UXM520). I generally use vanilla C for anything performance bound, but am increasingly moving to Python for everything else. Linux of course includes an endless supply of useful tools and libraries which I take for granted--gnuplot proved quite handy for visualizing the Netflix runs, for instance.

I think like many people, I worked on the Netflix thing round the clock for a while at first (well, as best I could while hosteling around New Zealand) and then that tapered off in response to the lonely cries of my other projects. Development of this sort is my full-time endeavor, though, so it's just a question of what's on the front burner at the moment.

Incidentally, I have been spending a lot of time these days without a ready internet connection, which is a hassle in many respects but at the same time I think I'm far more productive for it. I catch myself regularly wanting to pop over and check for new email or

Slashdot articles, but since they're not there I just hunker down and keep on with what I'm doing. And what I've realized after a bit of this is that it always happens coincidentally when I get to the really hard parts--especially when I get to a path I know I need to go down but which I've been down many times before without success. With ready distractions, it's possible to avoid these dark alleys essentially indefinitely. But when the excuses are all gone, you just suck it up and go for it, and eventually you work through it and have your moment of glee, and it's all worth it. And then you start the game with the next set of dark alleys--but hey, that's progress!

GPS: 5) What motivated you to share you code and ideas?

Simon Funk:

> *There's no limit to what you can accomplish if you don't care who gets the credit.* - anonymous

The bottom line is, life is short and I want to see things happen. I know for many this competition is between teams, but to me it's just part of a grander competition with time. The way I see it, there are a lot of pieces that need to fall into place to make some really fun and interesting things happen within our lifetimes, and this contest touches on one of those pieces. People may be viewing SVD as just a neat mathematical trick for minimizing RMSE, but when you take a step back there are a lot of interesting ties with other learning algorithms for pattern recognition, Bayesian inference and the like.

For instance, with this criss-cross Hebbian learning law where the activation from one set of nodes controls the plasticity of the other, just by changing the way the activations are normalized within each group you can shift from minimizing RMSE to maximizing likelihood. In other words, nearly the same algorithm will do Bayesian learning. And of course one has to wonder if something similar isn't going on in the brain, in the cortex specifically, where bottom-up and top-down connections converge in a sort of 69 symmetry reminiscent of this relationship. In short, there are some general principles here which I hope will seed some ideas and further progress in the wider domain.

GPS: 6) If you win the Netflix prize, what will it enable you to do?

Simon Funk: I feel like one of those lottery winners in saying this, but: I'd just keep doing what I'm already doing, but with a bigger buffer. Even with all the traveling I do, by the end of the year my budget typically adds up to about $15K. If you work one year in the bay area as a software engineer and save wisely, you're looking at two or three years off. If you spend those years continuing to work just as hard for yourself as you would have for someone else... Eventually you'll have something to sell or at least apply as a consultant and it becomes a self-sustaining cycle of gainful unemployment -- much like being employed but where you're working on exactly what you want most of the time. In a sense, I hope someone else wins on the off chance that it increases the ranks of AI entrepreneurs.

GPS: 7) What other interesting projects you would like to do? What would you like to achieve in 10 years?

Simon Funk: Most of my work is spiraling in on building a truly sentient machine. That's the main thing I hope to see happen within my lifetime, and I'm doing whatever I can to nudge it along. That pursuit inevitably spawns a lot of fun side projects, like the Netflix thing. I'm also working on a web site of sorts that incorporates AI behind the scenes in some unexpected places (besides the obvious collaborative filtering), and a new programming language/paradigm that tries to map more closely to how we humans think about and communicate problems and goals.

More centrally, I'm working on a model of human vision and potentially image synthesis--take the same process that allows you to close your eyes and imagine a scene, and add a "save as jpeg" feature. I definitely have more goals than there are hours in the day (the hardest part is to choose few enough that they stand a chance to get done), so I'm always looking for like-minded collaborators...

2. ACKNOWLEDGMENTS

I thank Vincent DiCarlo for his help in arranging this interview.

A first version of this interview appeared in KDnuggets, www.kdnuggets.com/news/2007/n08/3i.html

3. NOTES and REFERENCES

[1] since this interview was taken, another Netflix prize leader - ML at U Toronto team has published a paper describing their approach at www.cs.toronto.edu/~rsalakhu/mltoronto.html , but not in as much detail.

[2] http://sifter.org/~simon/journal/20061211.html

About the author:

Gregory Piatetsky-Shapiro is the Editor of KDnuggets and Chair of ACM SIGKDD (www.kdnuggets.com/gps.html).