EPISODE 50

[0:00:11.4] SC2: Hello and welcome to another episode of TWiML Talk, the podcast where I interview interesting people doing interesting things in machine learning and artificial intelligence. I'm your host, Sam Charrington.

This past week the conference finally came to me. Over the weekend, the great not so little anymore, Strings Loop Conference, graced downtown St. Louis. I got a chance to meet with a bunch of the speakers, including Soumith Chintala of Facebook, Allison Parrish of NYU and Sam Ritchie of Stripe. I had a ton of fun and I can't wait to share some of these great interviews from the conference.

Before we move on to the show, speaking of conferences, we're going into conference giveaway mode for a few days. Next week, October 10th through 11th, I'll be in Montréal for the Rework Deep Learning Summit and one lucky listener will get a chance to join me there. Entering the contest is simple, just head on over to twimlai.com/dlssummit and choose any of up to four methods of entry and, viola! There are only four ways to enter this time around, by sharing the contest with friends each participant can get up to 14 entries. This giveaway will only be open until noon central time on Wednesday the fourth, so make sure you get your entries in ASAP. Good luck.

This week I'd like to introduce a new sponsor, Nexosis, and thank them for sponsoring this week's show. Nexosis is a company of developers focused on providing easy access to machine learning. The Nexosis machine learning API meets developers where they're at regardless of their mastery of data science so they can start coding up predictive applications today in their preferred programming language. It's as simple as loading your data and selecting the type of problem you want to solve. Their automated platform trains and selects the best model fit for your data and then outputs predictions. Get your free API key and discover how to start leveraging machine learning in your next project at nexosis.com/twiml. That's N-E-X-O-S-I-S.com/twiml. Head on over, check them out and be sure to let them know who sent you.

Finally, before we dive into the show, a reminder about the upcoming TWiML Online Meet Up. On Wednesday, October 18th at 3 p.m. Pacific Time, we'll discuss the paper visual attribute

transfer through deep image analogy by Jing Liao and others from Microsoft Research. The discussion will be led by DuncanStothers. Thanks, Duncan.

To join the Meet Up or to catch up on what you missed from the first two meet ups, visit twimlai.com/meetup.

As you all know, a few weeks ago, I spent some time in San Francisco at the Artificial Intelligence Conference by O'Reilly and Intel Nervana. While I was there, I had just enough time to sneak away and catch up with Scott Clark, cofounder and CEO of SigOpt, a company whose software is focused on automatically tuning your models parameters through Bayesian optimization. We dive pretty deeply into how they do that through the course of this discussion.

I had a great time and learned a ton, but be forewarned. This is definitely a nerd-alert show. Without further ado, on to the show.

[INTERVIEW]

[0:03:52.0] SC1: Alright. Hey, everyone. I am here with Scott Clark. Scott is the founder and CEO of a company called SigOpt and he was gracious enough to spend some time with me this morning to talk about his background, the company, and the topic that I am very interested in learning more about; Bayesian optimization. We're sitting in his office in San Francisco. I happen to be in town for the AI Conference and I'm really looking forward to this interview.

Welcome, Scott.

[0:04:23.4] SC2: Thank you so much. Really looking forward to it as well.

[0:04:25.4] SC1: Awesome. Awesome. Let's just jump right in and have you tell us a little bit about your background and how you got involved in machine learning.

[0:04:33.8] SC2: Definitely. I first got really excited about this while I was in grad school. I was pursuing a Ph.D. in applied math at Cornell University.

[0:04:42.8] SC1: Go Upstate New York.

[0:04:43.7] SC2: Yeah, exactly. Cornell is great because there's not a lot to do and it's super bad weather all the times, so you just focused on studying and you graduate as soon as possible.

[0:04:51.9] SC1: I went to RPI undergrad, also Upstate New York, and had the same experience.

[0:04:56.1] SC2: Nice. I highly recommend it for efficient degrees.

[0:05:00.0] SC1: RPI had the added advantage of it was usually skewed towards male students, and so there were even less distractions.

[0:05:08.8] SC2: Fair enough. That's excellent. Basically, I was applying math to a variety of different things. One of the focuses of my degree was bioinformatics. I had a fellowship from the Department of Energy. The promo is trying to attack was genome assembly. You can think of this as trying to solve a jigsaw puzzle on a super computer. Basically, we have a bunch of the DNA and we have to reassemble it into some genome, and the Department of Energy cares about this, because if you know the genome, it might be a path where there's more efficient biofuels or something like that.

The problem was also tunable mobs of levers with these various systems and we had to configure those to get the best possible performance out of them.

[0:05:51.0] SC1: And you are the grunt in grad school to tune all these levers.

[0:05:54.3] SC2: We jokingly call this graduate student descent. The idea being we just need to get to the best configuration, and it doesn't matter how you get there. The standard way to attack this problem in a lot of different fields —

[0:06:04.7] SC1: You got to get your fee value for your paper.

[0:06:07.3] SC2: Something like that. Academic incentives are a completely different topic.

[0:06:13.1] SC1: I just thought I'd toss that in.

[0:06:15.2] SC2: Fair enough. The idea is there's a couple standard ways people go about attacking a problem like this. You could try to brute force the problem. Just lay down a grid of all possible options for every configuration and try them all. This was intractable for us because it took 24 hours on a government supercomputer every single time we wanted to try a single configuration.

Randomized search has become very popular, especially in the deep learning literature for trying to come up with different configurations or hyperparameters and architectures and things like that. It turns out much more efficient than grid search, but this is still trying to climb a mountain by jumping out of an airplane and hoping you land at the peak. Not necessarily the most intuitive way to go about optimizing something.

[0:06:56.9] SC1: A lot of the different algorithms we use randomized initialization. That's different from randomized search.

[0:07:03.1] SC2: Correct. When you're building a neural network, you might use randomized initialization on the individual weights and then you use some sort stochastic gradient descent optimizer within that underlying system. This is more of a black box parameter optimization problem I'm talking about where we're not introspective the underlying model, but just tuning the higher level configuration parameters.

Some of those configuration parameters might have to do with that random initialization or the stochastic descent parameters or something like that. You definitely need to be able to bootstrap efficiently from no data, but doing purely randomized search is not necessarily the most efficient thing you can do.

[0:07:42.9] SC1: Maybe before we move on, since I think we're going to be spending a lot of time talking about hyperparameter optimization here, maybe dig into grid search a little bit more

so that we're all starting from the same place. Basically, as I understand it, the idea is you've got some set of hyperparameters, those form and n-dimensional — Not a cube, but —

[0:08:08.8] SC2: Lattice.

[0:08:09.7] SC1: A lattice. Thank you. Grid search is basically systematically going from point to point, like if you were searching for someone in a forest, you'd kind of form a grid and kind of attack all those points. Random is you're basically picking points and the idea statistically if you pick enough points you'll gets some level of coverage of all of the combinations of these hyperparameters.

[0:08:35.0] SC2: Exactly. Back to your searching in a forest analogy, this is, yeah, jumping out of a helicopter and seeing if the person is there. Then being back at it [inaudible 0:08:41.3] and continuing to do that over and over again.

[0:08:43.3] SC1: That's random search.

[0:08:44.0] SC2: That's randomized search. Another popular method is just manual to me, trying to do this in your head. In the forest example when there's only two dimensions, you might have a lot of intuition about maybe the person is going to be up on a hill or something like that. It can actually be somewhat effective. Once you start to look at 20 dimensional problems, a lot of human intuition starts to break down and you might not be able to have some of that expert knowledge in the searching for a human in a forest setting, how to set stochastic gradient descent parameters and a number of hidden layers and lowering rates and all these sorts of things. It starts to get very convoluted very quickly. Manual search, while it can be effective to kind of resolve very localized solutions, is not a great global optimization strategy.

[0:09:28.1] SC1: For the typical model that you are seeing, how many hyperparameters are there?

[0:09:35.0] SC2: Yeah. It really depends on the underlying system. Something simple like a random forest might only have a couple that you care about, number of trees, number of samples you need to split in nodes, something like that. As you start to advanced, maybe

integrating boosting methods and all of a sudden you have learning rates and other sorts of parameters you can tune. Once you get into the deep learning and reinforcement learning regimes, there can be dozens of individual parameters, especially if you start to think of this system as a whole. When you're doing an NLP or a computer vision type problem, all of a sudden you have different ways you can parameterize the data as well. By looking at that system in its entirety, all of a sudden there can be dozens of parameters and something that grows exponentially like a grid search is completely intractable. The human manual intuition starts to break down and randomized search is just too slow to lock into a reasonable solution.

[0:10:27.2] SC1: Can you give an example of, in the case of NLP, how the way you look at the dataset changes and increases your permanent space?

[0:10:36.5] SC2: Yeah. How you tokenize the text itself. Do you look at different in-ground sizes? The idea being to do you look at one word at a time, pairs of words, triples of words? Do you maybe do different thresholds for the frequency within the corpus itself? Maybe cut out words like the because they're too common, and then also cut out words like Bonanza, because they're too rare. You can kind of change the actual feature representation itself before you even feed it into the machine learning algorithm, but these are all tunable mobs and levers.

[0:11:07.9] SC1: Got it. You were stuck in grad school, again, twiddling these levers and as all innovation happens, you thought, "There's got to be a better way."

[0:11:18.9] SC2: Exactly. I went around the department and found that this was a very common problem. People in for machine learning, people in financial engineering. Everybody were building these expert systems, but they needed to be fine-tuned, but everybody was using this kind of standard techniques. I expanded my search outside the department and eventually found who would become my Ph.D. advisor in the operations research field.

They've been attacking this problem for decades. If you have a time-consuming and expensive sample system, how do you most efficiently get to the best configuration? This crops up if you're tuning a particle accelerator. It crops up if you're trying to decide where to place a gold mine, which is where some of the original research came from in the 50s, but it maps extremely well on to a wide variety of computational problems.

You have some input that comes in. Some output they care about. How do you get to the best output and as few input attempts as possible? I started working in this field of optimal learning as it's called in operations research or sequential model based optimization or Bayesian optimization. A lot of fields have different names for it, but the idea is how do you do this as efficiently as you can?

Ended up pivoting my Ph.D. towards working on this problem and ended up being one of the chapters in my thesis. After graduating I realized that a lot of different people and a lot of industries have this issue. I spent 2 1/2 years at yelp working on that advertising team, applying these same techniques to help do more perform into advertising.

The idea being if you think about it mathematically, an advertising system is very similar to a genome assembly system, and so far as a lot of experts spent a lot of time building something. There's a bunch of inputs and there's an output you care about. In genome assemblies, it's better papers, because you get a better genome. In advertising system, a bunch of money comes out the other end.

[0:13:02.7] SC1: Clearly, there are tons of problems that fit that general scenario [inaudible 0:13:04.7].

[0:13:05.4] SC2: Exactly.

[0:13:07.1] SC1: You started SigOpt. How long you been at it here?

[0:13:12.0] SC2: Yeah, immediately after Yelp, started SigOpt about three years ago. Went through Y-Combinator and went through 15. Have raised a few rounds of funding, most recently a series A led by Andreessen Horowitz and now we're 16 people in San Francisco.

[0:13:25.5] SC1: Okay. That sounded like a steamboat or something. Or like a fog horn.

[0:13:31.1] SC2: I swear it's not normally this bad.

[0:13:31.8] SC1: So I guess I want to kind of jump in to the main course of this interview, which is around this Bayesian optimization. Walk me through the way folks like Pedro Dominguez will talk about the Bayesian's is like this one tribe within machine learning as supposed to the others. Kind of walk me through — I guess what I'm trying to get at is like I've had a couple of conversations with folks about different aspects of like Bayesian program learning and other things, but I feel like there's still some ethos of like what it means to be kind of Bayesian and think about things from that perspective that we haven't fully captured on the podcast. If we can start there and then get to the optimization that will be super cool.

[0:14:25.2] SC2: Yeah, definitely. The way a lot of those other techniques work, like grid search or random search, is there's no learning happening. I think that's one of the major differences between kind of the Bayesian optimization approach or the Bayesian approach to this problem and some of those more traditional techniques.

The idea being every single time I evaluate this underlying machine learning pipeline or whatever it is, it's extremely time-consuming and expensive. I want to be able to leverage that data to decide what to do next.

A lot of the Bayesian methods rely on this concept of trading off exploration versus exploitation. We want to be able to learn as much as we can about that underlying response surface, how it varies, how all the parameters interact, over what length scales? How certain we are about specific configurations and how well they'll perform and learn about that while also exploiting localized information to drive you to better results.

By constantly trading off these two facets we're able to exponentially faster than something like an exhaustive grid search arrive at better solutions. The main difference here is the fact that we're learning from the past and using that to influence what we do in future.

[0:15:38.4] SC1: Now when I think about this kind of explore-exploit tradeoff, one of the things that jumps to mind for me is reinforcement learning. Does that come in in play here or maybe less so because the environment itself, the problem itself doesn't necessarily change in response to the inputs?

[0:15:57.4] SC2: The underlying system can change pretty dramatically. You can think of this as this larger system that fits around any underlying pipeline. That could be reinforcement learning pipeline. It could be just a standard deep learning or it could be something as simple as a logistic regression or a random forest.

You can think about the fact that every single time we try a new configuration we want to observe some sort of output at the ends that the user defines. It could be something simple, like accuracy. It could be the sharp ratio of a back test of an algorithmic trading strategy or whatever it may be. We use that to kind of influence what we do next. You can think of this as kind of reinforcement loop as a whole over that entire system, but we're agnostic to what the underlying method is.

[0:16:41.2] SC1: I get that. The underlying method could be reinforcement learning or any number of other things, but it also sounds — I guess what I was asking was are you or could you do reinforcement learning at the top level to optimize the thing that you're optimizing, which could be reinforcement learning as well. Reinforcement learning on the hyperparameter space as supposed to the actual model itself.

[0:17:08.5] SC2: Yeah, definitely. There're a lot of different approaches to this underlying problem. There're a lot of very cool papers that are at all the-top machine learning conferences for attacking this. The way that we attack it is via this concept of sequential model based optimization. This is a very Bayesian approach, and the idea is we're sequentially learning as much as we can about this underlying system. Once again, using the history to decide what to do in the future.

It's model based in the sense that we're building up different surrogate models for how we think individual configurations are going to respond when we actually sample the underlying system. We can use various different things here, like [inaudible 0:17:48.4] processes or other kind of Bayesian regression type systems. We want to be able to say, "Given what we think is going to happen, how do we sample as efficiently as possible?" Then we want to say, "What do we think is going to improve an expectation the most? What's the highest probability of improvement in terms of that new configuration to suggest?" Then that loops back into the underlying system

after you sample it and we learn, update the posterior of these individual surrogate methods, optimize on them and repeat that entire process.

[0:18:23.9] SC1: How do you get to the proposed model for the model base piece of this?

[0:18:32.2] SC2: In general, in Bayesian optimization, usually you pick a specific type of model and go from there. Some of the open source work I did at Yelp, it was very cut and dry. Use a Gaussian process, use expected improvement to optimize and go through kind of extremely sequentially. This is very similar to Spearmint and other popular library.

[0:18:52.3] SC1: Say that one again?

[0:18:53.1] SC2: Spearmint. It was an open source library out of Harvard. Very similar to the metric optimization engine or MO, which I wrote at Yelp. Also similar to like [inaudible 0:19:01.4], which is kind of a more recent one. This is kind of the bread-and-butter Bayesian optimization approach, Gaussian processes is expected improvement.

What SigOpt represents though is this ensemble based approach. Different surrogate models, different acquisition functions, different covariance kernels learning how the parameters interact as well as not just kind of that standard build, a single sequential surrogate model based approach, but really taking all of these different optimizers and optimizing and making it automatic.

You can select something ahead of time, because you know you want to take a very specific approach, or you can take the more generalized approach and say, “We’re not necessarily going to say we’re going to use this specific surrogate model. We want to learn along the way with the best possible thing for that underlying system that we’re optimizing.”

[0:19:53.5] SC1: To take a step back, you are in the former case where you're picking a model, a specific model. Let's say we’re assuming a Gaussian distribution, then basically we’ve got this hyper-parameter space. I’m trying to get at how — The parameters of your Gaussian distribution that’ll be your mean and your standard deviation. How are we — What's the process for identifying those that is then — That we’re doing sequentially?

[0:20:27.9] SC2: Got you! The way that a Gaussian process works is that it's assuming that the response of the underlying system that we're sampling is going to be Gaussian distributed at any given pinot. It's not a single Gaussian distribution or something similar to like a Gaussian mixture model. What it actually is is an infinite number of potential Gaussian responses for every potential input. Then the way the Gaussian processes are analytically defined, once you start the sample underlying points, you can explicitly build up what that distribution is at sample points or on sample points.

The main thing that controls this is what's called a covariance kernel, and what that is is how much information do I get from sampling point A about some other point B? Does it decay exponentially? Is there some sort of high variance or noise associated with it? What are the length scales over which all the different parameters interact? This becomes doubly complicated once you start to look at heterogeneous configuration spaces with integers and continuous variables and categorical variables and things like that.

[0:21:35.5] SC1: This covariance matrix, is this something that you're learning as part of the process? It's not something that you know a priority.

[0:21:44.7] SC2: Exactly. You can set it a priority, but you can also learn as you go. There are tunable parameters around these covariance kernels. Yeah, it turtles all the way down. The idea here is once you can analytically define, this is maybe a surrogate function I may use. I use a Gaussian process. Here's a specific class of covariance kernels, like an ARD kernel or something like. Then you can explicitly say, "Okay. How good is the fit given what I've observed so far?"

Because you're defining the system analytically and you've effectively mapped the problem from, it's extremely sparse, time-consuming and expensive, underlying system that you're sampling and now you've mapped it over to this surrogate space. You can start to throw kind of the kitchen sink of mathematics of the problem and use that to kind of optimize the end of the line covariance kernels, pick the correct ones, find the right surrogate functions and then ultimately leverage that information to decide what's the point that has the highest probability of improvement or expected improvement or whatever it may be.

[0:22:50.6] SC1: Is the surrogate space, in this case, the covariance kernel or kind of this vector, this infinite vector of the distribution?

[0:23:00.7] SC2: The covariance kernel defines that infinite vector, or at that functional distribution. There's two ways to think about Gaussian processes.

[0:23:08.9] SC1: Your covariance kernel is infinite by infinite dimensions or something on that order?

[0:23:14.6] SC2: I mean it can —

[0:23:16.2] SC1: How do you — Is part of the goal to kind of constrain the dimensionality of this covariance kernel?

[0:23:22.3] SC2: The covariance kernel itself will take in inputs in the configuration space and basically say how much covariance can I expect between these two points. It does map into a real number. Technically for various types of covariance kernels, there are these tunable parameters that are continuous. Technically, yes, there's an infinite number of different ways you can parameterize that. What we're able to do is, say, given what we've observed so far, what's the most likely parameterization, or what's a distribution of likely parameterizations and leverage that to decide, "Okay. This is what we think is a reasonable surrogate function." Then once again do that across a wide variety of them.

[0:24:05.5] SC1: Okay. I'm still not fully getting where the infinite distributions come in.

[0:24:13.2] SC2: Yeah. There's two ways to think about a Gaussian process. One is from the point-wise perspective. The idea is at every single point we're going to assume the response from this underlying system that we're sampling. It's going to be Gaussian distribution. Every single potential configuration has a different potential Gaussian response to it. There's some —

[0:24:33.4] SC1: Meaning, so you've got and input point and then you've got the space of configurations and each of those configurations translates this input point to a different distribution.

[0:24:48.6] SC2: The input point is a potential configuration. Maybe I'll take a step back and do an explicit example here. Let's say we're tuning some neural network and we want to find the ultimate learning rate. Maybe initially we try something like just .5 or something like that and we get a response back. Okay, and we're optimizing for the accuracy of a fraud detection pipeline. We'll be, "Okay. We get .7 cross validated AUC. That looks alright."

The thing that we're optimizing for is our learning rate and the input is — We're not talking about inputs to our neural network and outputs to our neural network. We're talking about an aggregate, the error.

The inputs are — We're going to be tuning this machine learning pipeline. At this high meta-optimization layer, we're going to be saying, "Okay. We're going to put in a learning rate and then we're going to go through the training and cross-validation and all sorts of things and come up with some metrics that we care about." Maybe cross-validate it to AUC. Our goals is to find the learning rate that tunes this entire pipeline in such a way that it maximizes that output.

The way that this works in the sequential model-based optimization framework is, "Okay. We sampled .5 learning rate and got .7 out as the result." Maybe there's a little bit of uncertainty associated with that. Then let's say we want to model what we think is going to happen if we try .6. We have a little bit of information because we've already sampled .5. What we do is we build up this Gaussian process that says, "Okay. I'm pretty sure that it's going to pass near this point that I've already sampled, but then maybe the information decays pretty rapidly." I expect to see maybe .6 plus or minus .1 if I were to sample point further away from it."

What you can think of is every potential input learning rate to tune this pipeline has its own Gaussian response that we're expecting. It has its own mean. It has its own variance. We can explicitly build that up once we define the covariance kernel. Of course, as you expand this out into more dimensions.

[0:27:02.7] SC1: In this example we're talking about what is a covariance kernel look like.

[0:27:07.0] SC2: Yeah. We would explicitly set the covariance kernel, like an ARD kernel that says, "Okay. We're expecting some sort of like squared exponential decay of this kind of information from sampling these different points.

[0:27:21.4] SC1: And so is the covariance kernel, again, in this particular case, it's going to describe the relationship between the learning rate and the output.

[0:27:31.3] SC2: It's going to describe the relationship between individual samples of that learning rate. Does that vary where we expect wildly different results after .01 increments, or is it .1 increments? Do we expect to be an extremely noisy response, although we expect it to be fairly well-behaved? There's various different parameters of this covariance kernel that basically say, "How much information effectively do I get after sampling point A about some other point B?"

[0:28:04.1] SC1: Is the dimensionality of the covariance kernel fixed when we start, or does it increase in dimensionality as we sample?

[0:28:13.5] SC2: It takes an input, which is the actual configurations. In this case it would just be a one-dimensional, just the learning rate, but you could imagine that's extending this out. It takes in a vector, which is specific configuration or two vectors actually and says, "Okay. How much covariance is there between these two points? These two potential configurations?"

That being said you can parameterize that covariance kernel in different ways depending on which specific kernel you've picked. In something like an ARD kernel, which is this squared exponential drop off, there's various length scales that you can tune. Maybe we know a learning rate —

[0:28:50.4] SC1: How fast the drop off is, that kind of thing.

[0:28:51.7] SC2: Yeah. Does it vary over.1, but then something like the number of hidden layers might vary over orders of magnitude larger? A hundred hidden layers is very similar to 101 but very different than 200.

[0:29:05.0] SC1: I'm still not sure that I'm very clear on the kernel. In the specific example, the dimensionality of the kernel is one by one? Is that a scaler or —

[0:29:15.9] SC2: Yes. It takes in a single value, so that's just the learning rate.

[0:29:22.6] SC1: I guess I'm thinking of it as a matrix. Is it a function or is it something — Should I not be thinking of it as a matrix?

[0:29:31.6] SC2: Yes. You can define it as a matrix or it's every point — The pair-wise covariance of every point you've sampled so far.

[0:29:39.9] SC1: As you sample the dimensionality of this thing is growing.

[0:29:43.7] SC2: Of the underlying covariance matrix, but the underlying covariance function is just a function. There's no kind of dimensionality associated with it.

[0:29:51.1] SC1: Okay.

[0:29:52.2] SC2: Basically, if I've sampled 10 different points, that I could have a 10 by 10 matrix, which is the covariance matrix where every single actual instance inside that matrix is how does .7covary with .3, or whatever it may be. This, as a whole, helps us define the Gaussian process, which then gives us this stochastic surrogate function for what we think is going to happen if we sample outside of the points that we've already explicitly observed.

[0:30:23.9] SC1: Okay. It does that by way of defining the kernel. How do we get from the matrix to the kernel? Is that done explicitly?

[0:30:32.5] SC2: It's the other way around. You start with the kernel and then the kernel defines the matrix. Every single individual value within that matrix is defined as —

[0:30:43.8] SC1: I got it. We're specifying the kernel. In this case, you said ADR is —

[0:30:47.6] SC2: Yeah, ARD.

[0:30:48.0] SC1: ARD. What does ARD stand for?

[0:30:52.2] SC2: I haven't said that? I'm blanking on that all of a sudden.

[0:30:54.4] SC1: It's a squared Gaussian falloff.

[0:30:56.3] SC2: Yeah.

[0:30:57.3] SC1: What's now unclear for me is if you've picked a sample in your input space and you've run your underlying process and you have an output value from that sample, is the covariance kernel used to build up what you expected to see and then you push that all through and you get what you actually saw?

[0:31:18.9] SC2: Then you can update the covariance kernel, and then that covariance matrix gets one more row and one more column, because not we have how this new points varies with all of the previously observed points. Then we can use that to update our Gaussian process, and now we have this new posterior result that we can use to decide what we sample next.

What we're doing is we're not just kind of doing naïve optimization on that Gaussian process response itself. We don't just want to find the point with highest mean or something like that. What we want to do is apply an acquisition function to it and say, "Given this is what I think is going to happen if I sample any of these potential input points, how do I find the point with the highest expected improvement or the highest probability improvement, or which one is going to give me the most knowledge about the eventual optimized, the knowledge gradient method?

[0:32:15.3] SC1: And so acquisition function is a new term that you just introduced. Is that something that is model based, like the covariance kernel is model based or on this ARD, or do you pick a model that you use for your acquisition function as well?

[0:32:30.0] SC2: Yeah. This is the optimization part of it. The sequential part of sequential model-based optimization is leveraging the history to build up these surrogate models.

[0:32:39.3] SC1: [inaudible 0:32:39.5] The covariance kernel and keeping it updated and all that stuff.

[0:32:42.2] SC2: Yes. The model base part is actually deciding, "Okay. This is what we think the response is going to be in these un-sampled configurations. That's the Gaussian process. Then the optimization component is given that surrogate model, what do we actually optimize for sampling next before we repeat this entire process?

[0:33:03.7] SC1: That particular piece is really focused on you've got this massive potential state space for your hyperparameters. How do we choose a sample path through the hyperparameter space that minimizes basically wasting time and not adding information to our process?

[0:33:22.8] SC2: Exactly. This is what really controls that explore-exploit tradeoff. A popular acquisition function is expected improvement, and that is basically how much do I think I'm going to beat the best thing I've seen so far by? I've seen a pretty good AUC in my fraud detection pipeline, now all the sudden one I want to be able to do it as well as possible beyond that. We're playing King at the Hill effectively.

Another popular one that's kind of maybe a little bit more intuitive to grasp is probability of improvement. If I were to sample this un-sampled point, what's the probability that I beat the best thing I've seen so far? These have different exploration exploitation trade-offs in so far as probability of improvement might be a little bit more conservative, like we're going to kind of keep edging it up slowly. Whereas expected improvement kind of takes the magnitude of the game into account. It might try something far away, because it thinks there could be something great that it has just never seen before.

[0:34:18.9] SC1: Yeah. Are there other common examples?

[0:34:23.1] SC2: Yes. Another one — Unfortunately they get a little bit more complicated internalize. Another popular one is knowledge gradient. This is what my Ph.D. advisor worked on during his Ph.D. The idea is —

[0:34:34.1] SC1: I'm imagining from the name, like that's kind of based on information theory and like how much we're going to learn by checking this point.

[0:34:41.0] SC2: Exactly. The goal is to learn as much as we can about that eventual best point. There's more information theoretic acquisition function. Then you can kind of define anything that you want with a goal of eventually getting to these best once. These are probably the three most popular, but you could imagine doing composites of this or some sort of like upper-confidence bound base acquisition function. The idea is you want it to as efficiently as possible trade-off exploration and exploitation, because learning about that underlying system and how it performs and things like that is important. But at the end of the day you just want the best performing model.

[0:35:18.1] SC1: I think it turtles all the way down, strikes me as app. You've got hyperparameters for your model. You've got hyperparameters for your pipeline, and then you've got hyperparameters for your optimization system. Presumably, I'm imagining that you are also trying to optimize hyperparameters at that top layer for your optimization system as well.

[0:35:40.8] SC2: This is exactly why SigOpt exists, because there's some incredible research out there. A lot of members of our team have contributed to the academic research in a lot of the open source out there. There's a lot of promise that Bayesian optimization has. Unfortunately, a lot of expert time is wasted optimizing the optimizer, figuring out the best way to tune all of these turtles all the way down. I think that's one of the places where at least the open source that I released, the metric optimization engine, even though it was very popular no GitHub, it kind of failed to deliver on that promise, because it required an expert to sit and fine-tune all these different things.

The goal of a company like SigOpt is can we optimize the optimizer for you and create this automatic ensemble that makes all of these trade-offs so that you as an expert can focus on fraud detection and will focus on black box optimization for you.

[0:36:33.1] SC1: Okay. We've described a bunch of different kind of variance in this process. Are there specific in variance for SigOpt and your process, like for example basing everything on a Bayesian process, that's one way to doing this? Is the product based around that, and what other kind of invariance are there in the way you approach this?

[0:37:00.7] SC2: Yeah. At the very highest level, we're just black box optimization. There's inputs to a system. There's an output or set of outputs that we want to optimize and we're going to try to come up with the best set of inputs.

Bayesian optimization is an extremely efficient way to do this, especially when it's time-consuming and expensive to sample that underlying system. There's lots of different variants of Bayesian optimization. Instead of using like a Gaussian process, we could use a Bayesian neural network for the underlying surrogate function.

Instead of using Bayesian optimization, we can use a genetic algorithm, particles form or simulated annealing or even just a convex gradient-based method. The idea being SigOpt takes care of that optimization of the optimizer and automatically selects the best one for you. Most of our methods or almost all of our methods are Bayesian in nature, but we're not constrained to that necessarily.

[0:37:52.9] SC1: Yeah, I guess that was the question that I was trying to get at. How far do you go? Do you also now or envision a future where cause you're providing this black box capability, you may do the Bayesian optimization, but also sample or test the results that you'd get from particles forms and other types of methods.

[0:38:19.9] SC2: Definitely. In-house, we've built this very robust evaluation framework for deciding whether or not specific algorithms fair well in different contexts. This is what we use when we integrate a new paper and want to make sure that with high statistical confidence it actually outperforms what we're currently doing. We use this as kind of our internal metric for deciding what to do. We're agnostics to the underlying methods. We just want the best possible thing for our customers. It turns out for the types of problems that we're attacking, Bayesian optimization is an incredibly good fit and it's kind of underutilized because it's so difficult to get

up and running and optimized. We have and we'll continue to employ whatever the best is for the problems that we're attacking.

Because we define this barrier in this way, where it's just black box optimization, the underlying system as a black box to us, but we're also a black box to your customers. This allows us to kind of hot-swap in the best possible technique to solve their problem and not be constrained in that way.

[0:39:21.3] SC1: Okay. Cool. Can you talk a little bit about the model evaluation framework that you've built?

[0:39:26.3] SC2: Yeah. There's some ISIMA workshop papers from 2016 to go into quite a bit more detail that are available on our website. The idea is I've just told you that we have an optimization framework that can solve any kind of underlying black box function. The first response should be, "How do I know whether or not it's working?"

Internal, we built up the system where kind of traditionally to publish papers, and I'm guilty of doing this, is you would come up with some strategy. Pick 3 to 6 of your favorite functions, show that you can outperform some specific techniques on those functions, publish a paper, rinse and repeat.

When we built the up internally, we took the superset of all those different functions from the academic literature. We took functions that look similar to our customer's data. We took a bunch of open machine learning datasets and strategies. We basically piled them altogether. Instead of comparing against three or four different response services, now we're looking at hundreds or thousands of them.

In addition to that, we wanted to make sure against all of these different open-source methods and against all these other kind of different global optimization strategies, that we could very robustly outperform them. What we do in the internal evaluation framework it we independently optimize these hundreds of different pathological and real-world problems many times with SigOps and many times with another method. That other method might be just a new version of SigOpt. Then with high statistical confidence we can say which one got to the best value

fastest? Which one got to the ultimate best result? Which one was the most robust? It didn't have like an interquartile ranges are all above a specific value.

[0:41:04.8] SC1: It sounds like to draw an analogy from a software engineering, you built a regression testing framework for optimizers.

[0:41:10.6] SC2: Yes. We do use it for aggression testing. It's run nightly, but it's also a way to basically A-B test optimizers as well.

[0:41:18.0] SC1: Right. You're not using it to — To what extent are you using it to inform model choices, or I guess what I'm struggling a little bit with is — So you've got this heap of datasets and functions and things like that if you were trying to optimize across all of those, then you've got a least common denominator kind of problem or a local maxima or something like that.

[0:41:50.6] SC2: Yes. We do have to be weary that we don't over-fit to this dataset. That's definitely true. One thing that we found though is the reason why we built an ensemble based approach —

[0:41:59.3] SC1: Let me just poke at that. Is over fitting the right word for — What I'm thinking of is that if — It strikes me as the opposite of over fitting. Whereas like if I were to just look at — I don't really care about all these other data. I care about my problem. If you're optimizing for this kind of broad spectrum and I can outperform you by just focusing on my problem, I'd probably do that.

[0:42:24.2] SC2: Yeah. That makes a complete sense. I see where you're coming at here. This is why we take this ensemble based approach, because it turns out like the most popular approach doing Bayesian optimization, like Gaussian processes with ARD kernel, with expected improvement actually doesn't do super well in a wide variety of different contexts.

By sleuthing in the right tool for the job we can actually hit all of these different facets of different types of problems extremely well. That being said, the no free launch theorem in computer science still applies here in so far as if you do have expert knowledge about your underlying system and you build a bespoke optimizer to solve that one specific problem, you are going to

outperformed a general technique. That being said, you would have to repeat that for the next problem that you attack and the next one and the next one.

The idea is by having an ensemble of different optimizers we use the right one for specific context and then a different one for a different context, etc. Instead of having like the lowest common denominator, like you said, just the one-size-fits-all, what we're doing is actually putting in the right tool and automatically learning when we trade it off. When you're tuning a gradient boosted method, you're getting the right tool. But when you tune a neural network, it's still the same API and same interface, but you're getting the right optimizer.

[0:43:39.9] SC1: Got it. What I'm hearing is in response to my question, like a little bit of both. You've built this model evaluation framework because fundamentally you're not necessarily trying to outperform a handcrafted model that 50 Ph.Ds has spent five years developing, whatever. You're trying to build a system that can deliver good performance on, in general, what someone throws at it. You want to test it against a bunch of, "Hey, these are things that someone might throw at it," and make sure that you get good performance. The way that you do that is under the covers you're not just relying on one specific set of choices, but you're taking an ensemble approach and your optimizer can swap in and out different decisions to produce a result that's best.

[0:44:33.8] SC2: That's exactly it, because what we find more often than not is that people don't assign 50 Ph.Ds for five years for every single optimization problem they have. More often than not, they're using grid search, random search, manual tuning. Maybe an open source solution, maybe they have part of their team part-time working on an internal optimizer or something like that, and those are the things that we can vastly outperform. If you know it's convex and you have gradient information and you have a bunch of expert knowledge, like there is specific tools that you can use to get there and this is probably little heavy-handed to use in that situation.

More often than not doing, what we're doing is we're coming in and replacing these very exhaustive, very expensive, very domain expert intensive systems and we can generally outperform those to a high degree.

[0:45:20.5] SC1: I often like to think of the tool space in general, is like there's — For many enterprises, there's a such a huge potential opportunity to apply ML that their ability to staff up is far outpace by the opportunities. At a given staffing level, like you've got this choice. You can either like take only the biggest opportunity and apply all your resources to that in a very manual way or you can utilize tools that allow folks to be more effective and bite off some of the — It's like a lot of — I'll talk to folks and they would talk about it like we only go after homeruns versus base hits. It sounds like this is a tool for allowing people to both go after homeruns as well as try to increase their hit rate for base hits.

[0:46:15.3] SC2: Definitely. We're finding with a lot of the firms that we work with is how they differentiate themselves from the competitors is not by black box Bayesian optimization. It's by creating a great recommendation engine or a great algorithmic trading strategy. If you can hire five more Ph.Ds to work on that core differentiator or free up five Ph.Ds to do that and then just use SigOpt to tune it, they work very additively and hand-in-hand we can accelerate that time-to-market, accelerate the results getting to the best performance and all of these different things. I think more and more companies are becoming aware of this and using the right tool for the job. Why we write Tensor Flow when you can use it. Why write your own Bayesian optimizer when you can use a best in class easy REST API?

[0:46:59.9] SC1: Awesome. What's the best way for folks to learn more? I'm assuming the website?

[0:47:04.1] SC2: Yeah, sigopt.com or just contact as sigopt.com if you want to shoot us an email. We ran a complementary proof of concept pilot. We can throw these peer-reviewed papers at you to prove that we're as good as we say we are, but at the end of the day we want to prove it with their underlying models themselves. We can work with any enterprise, any underlying system, cloud agnostic, model agnostic. It's also free for students. If there are any people at universities or researchers at national labs or whatever it is listing the podcast, sigopt.com/edu gets you a enterprise account. I wasted way too much of my Ph.D. on this problem. Don't want to do that for anybody else.

[0:47:39.4] SC1: What about for folks that are interested in learning about the theoretical foundations of work? Where would you point them? Are there like three canonical papers or something like that that they should look for?

[0:47:49.1] SC2: Yeah. If you go sigopt.com/research, those are all of our papers. We also have a Bayesian optimization primer there that kind of goes into more detail about some of the things I said. Verbally, sometimes it's a little bit hard to describe Gaussian processes and things like. The math is there. The references for all those papers as well. That can kind of take you down the rabbit hole of all the different ways that this has been applies historically.

[0:48:10.3] SC1: Okay. Awesome. Thanks so much, Scott. It's been a great conversation and I've learned a ton.

[0:48:14.9] SC2: Excellent. Thank you so much. I really appreciate it.

[0:48:16.7] SC1: Thank you.

[END OF INTERVIEW]

[0:48:21.7] SC1: Alright everyone, that's our show more today. Thank you so much for listening, and of course, for your continued feedback and support. For more information on Scott and the topics covered in this episode head on over to twimlai.com/talk/50. Next week, on Tuesday and Wednesday, October 3rd and 4th, I'll be at the Gartner Symposium in Orlando where I'll be on a panel on how to get started with AI. If you'd like to meet up there please send me a shout.

The following week I'll be in Montréal for the Rework Deep Learning Summit and hope to be joined by at least one lucky listener. Remember to visit twimlai.com/dlsummit to enter. Contest ends at noon Central on October 4th.

Thanks again for listening, and catch you next time.

[END]