Module 12 Transcripts

Video 1: Introduction to module 12 [01:29]

This module is about Bayesian optimisation. Bayesian optimisation is about the optimisation of expensive to evaluate possibly black-box functions. This area is so important that we're going to centre our capstone project around this particular area. This is such an important area that many fields have invented or reinvented the ideas that are here. So, some synonyms that you might encounter are: surrogate modelling, response surface modelling, kriging, black-box optimisation, Bayesian optimisation, and Bayesian quadrature. These terms aren't exactly synonyms, but they are very, very closely related to one another and often have identical ideas in them.

So, this idea originated from a master's thesis from 1951, this is from Krige, he was at a university in South Africa and what he was interested in is mining. So, if you think about a mining problem, I have a very large earth surface, and I can go into one little Earth's surface, maybe with a drill, and I can try to figure out if under that particular point is the thing that I want to mine. Now, this is extremely expensive to evaluate because, of course I have to dig a great big hole in the ground, and whether or not there is what I want underneath the surface of the Earth, I have no idea maybe there'll be something else that I like under the surface of the earth that's close by, maybe not.

So, there's no special information about the function. So, what Krige thought about is, how do I develop a way of representing what's going on, when I can only evaluate points, but I have no other special information about this function? This idea was extended in 1964 to global optimisation. What happened here is that Kushner was interested in optimising functions where you can only evaluate the function itself, no extra information. What this diagram is showing is that we have this bimodal curve.

There are two maximum points, two local maximum, so the curve is going up and down and up again. And if I want to maximise this function, well, then I would hope I was developing a method that would evaluate the function more often near where the high points are, rather than near where the low points are. And indeed, that's exactly what this graph is showing and exactly what this method does. In the early 2000 the machine learners recognised that this particular approach was very, very useful for problems in machine learning, where you can evaluate function, you can do something like you can train a neural network, with a set of parameters.

But you don't know if you used a slightly different set of parameters what would happen then? So, there's this seminal book that came out in 2006 by Rasmussen and Williams. And what they call this is they call this Gaussian Processes. This is what we will call one of the tools that we use for Bayesian optimisation, this is not the term that a geologist would use, but we are after all machine learners. So, first off, let's talk about all the applications or some of the many applications where Bayesian optimisation is very, very useful. There's an application of making a robot walk, and what we have here is we have a robot that we can set on a table, and then we can set the different parameters so we can set say, how long the legs are and we can set how fast the gate is, and then we press go.

And the robot is going to walk across the table. Maybe maybe it'll just fall over. Maybe it'll walk one step and then fall over. Maybe it'll get across the whole table. What I want is I want to get the robot across the whole table just by tuning parameters. I don't have that many tries actually, because if you're like me, then you're going to end up breaking your robot pretty soon because some of your tries are going to be bad. So, basically, what you need is to be able to evaluate this function not too often and have the robot walk across the table in a nice way. Another application that authors have thought about is in sensor set selection. Here this is somehow related to the geology example where now we have the entire island of Great Britain, so it's a very big area, and we can place sensors where we wish to place the sensors. Now, this is again going to be an expensive to evaluate function, and again, I don't want to evaluate all that many times. Another application that I've looked at with my team is in tissue engineering. So, in tissue engineering they have these super expensive to evaluate functions. So, one experiment for these people takes them a month.

So, let's say you set up your experiment wrong at the beginning of the month you get to the end of the month, you might find out that you have basically no useful information, this is really discouraging. So, one of the things that my team looked at with a team of tissue engineers is basically how we can use this technology of Bayesian optimisation that we will discuss in this module to set up the experiments in a nice way. So, what I'm showing here is, I'm showing the reactor that the tissue engineers designed, inside the reactor there is this three dimensional scaffold where there are cells growing on the reactor.

So, so far, I've shown you a number of applications where Bayesian optimisation is successful. But what you should always ask when somebody shows you a whole bunch of examples that are successful, is okay, when is this not successful or what are the limitations and trade-offs associated with this method? First off, I want to talk about dimensionality because we're only evaluating this function, and it's expensive to evaluate. We're not getting any additional information about the direction of the parameters or something like this. This is the thing that's going to work in small dimensions, so think as a rough heuristic, less than or equal to 10.

There's a lot of active research in seeing can we consider functions that are much larger in dimension than 10? But think of it as dimension, less than or equal to 10 is effectively a solved problem. Another thing that people think about in terms of whether or not an application is going to be successful is, what are the variables or parameters that we're going to tune in Bayesian optimisation? So, I gave a number of examples in the geology example. You have the Earth, and the Earth is of course, a surface and you can drill that may be continuous places.

But then I gave the example of this reactor for tissue engineering. The tissue engineers don't have any reactor design available to them. They have specific reactor designs available to them, and so what we have there is we have so called categorical variables where we can choose individual reactor designs, but we maybe can't combine to or something like this. So, what we want is for the surrogate model to somehow recapitulate the underlying function. So, we have to ask ourselves about the tuning parameters, we have to ask ourselves are the underlying variables are they continuous? Are they categorical? And then we might want to ask ourselves about periodic function aspects.

So, for instance, if this had something to do with the tides or the phase of the moon, then we would want it to come back on itself. And we would want to design our underlying surrogate model to somehow also have a periodic aspect. Finally, there are some important things to think about with respect to numeric difficulties. So, here in Bayesian optimisation in this module, we will talk about a number of technologies, one of them calcium processes, that we'll talk about quite a bit, does have a challenge of inverting matrices, and I mean I know that sounds like something you know from linear algebra from a long time ago, but inverting matrices is not easy once you get to the size of matrices that needed to be inverted for this particular application. My top tip is to reformulate so that you start with a mean of zero, so that your prior belief about the function is that you have a mean of zero.

Video 2: Surrogate models [05:11]

In Bayesian optimisation, one of the most important decisions to make is what surrogate model to use. So, remember that we have parameters that may arise in machine learning models. When these parameters arise, where we may need to tune them. So, we could sometimes optimise out these parameters or hyperparameters, using something like maximum likelihood estimation. However, there are cases such as in transparency or interpretability, where we actually want to manage what are the values of the parameters or what values they can take. So, for instance, we might want to tune ourselves the depth of a tree for a decision tree, or the number of trees of decision trees.

When we're thinking about surrogate models for Bayesian optimisation, we would like the surrogate model to somehow match these parameters that are tunable and important. So, the first thing that we often use in surrogate modelling is what are called Gaussian processes. We alluded to those in the previous video. What a Gaussian process does, is that it generalises a normal distribution to an infinite number of dimensions. This is a really cool concept. Because Gaussian processes are so tightly tied with an infinite dimensional representation of a normal distribution, you get amazing statistical properties. So, the theoretical properties of Gaussian processes are great.

You can prove all sorts of nice things. You can handle continuous variables or parameters in a really nice kind of way. And because you have this nice normal distribution or infinite normal distribution that is underlying the function itself, you have automatic uncertainty estimates. So, basically if I have evaluated a point, I somehow have some sort of understanding that's based on the normal distribution as to what's going to happen in areas around that point. A challenge of this particular type of surrogate model is that you'll run into numerical difficulties. As the problems get bigger, you have to invert these really big matrices and that is numerically difficult.

Another commonly used surrogate model is so called regression trees. With regression trees these are similar to the decision trees that you have seen earlier in the course. You've got more numerical stability, and now you can handle categorical variables. The downside of regression trees is that there are no embedded uncertainty estimates, so there is no sort of thing that's built straight into the function itself that gives you some understanding of the uncertainty.

I'm giving a picture here of a way that we could embed uncertainty into regression trees. So, here what's going on is that we have a dotted, curved line that is the function that we are trying to model. That is being somewhat well modelled by a darker, bold line that is sort of it's either horizontal or it's vertical. Just like a tree is going to be either horizontal or vertical. That bold line that represents the tree, is basically recapitulating data points that have been sampled, those are the dots. And then we can add in this distance-based uncertainty metric, by for instance saying that the farther we are from a point that we have already evaluated, the less we are likely to trust the function.

So, I mentioned Gaussian processes, and I mentioned regression trees. Of course, these are not the only surrogate models that can be used. First off, there are many stochastic processes that can be used, for example, Dirichlet processes, Levy processes, etc. These have nice uncertainty characteristics. And then there are all sorts of other regression functions we can consider: linear functions, polynomial functions, spline, neural networks, etc. And then, in particular, the Gaussian processes, what we're showing in this particular image is the way that the Gaussian process itself is basically a function of functions, there's lots of functions inside of it.

And so, what's going on is that we have a dark, solid line that's showing where the mean of the Gaussian process is currently. And then, we have a bunch of data points that have been taken where now we sort of understand better what is the shape of the Gaussian process. And then we can take draws from the functions that are inside the Gaussian process.

Video 3: Representing exploration vs exploitation [03:10]

In Bayesian optimisation, one of the things that we're very interested in is, in sort of representing exploration, that is sort of maybe we should drill places that are very different than we have looked before. Or maybe we should go investigate, sort of reactor conditions that are very different than we've looked at before versus exploitation. Exploitation is of course, well, this worked well, so let's do something similar. So, how we're going to represent this mathematically is the topic of this video. First off, let's say that we're trying to recapitulate the function that is shown here as a solid line with some noise on either side.

So, what I'm assuming here is that we have a noisy function. I'm going to evaluate it, but I'm not necessarily going to get out the solid line. I might get out some error around the solid line that would be my noise. But, because this is an example of Bayesian optimisation, I do not have any information except for those dots. So, all I have is not the function itself or the line itself, it's just that I can query the function and I can get out an answer that's basically going to be the function with the noise somehow embedded.

So, this is what the Gaussian process is going to see, is just this collection of points that is not the function itself, but is where we evaluated the function. So, let's say that we're going to use as a surrogate model, we're going to use a Gaussian process. And what we said that the surrogate model was going to see is the points. What we want our surrogate model to do, is that we want it to develop a mean function, this is the dotted line that is going through those points that were the ones that were able to evaluate.

And we also want some prediction uncertainty. Here, I'm illustrating the variance as a lighter colour to the mean prediction. In particular, the example that I'm giving right here, this is a Gaussian process representation, a regression tree representation, or a neural network representation might look a little bit different. When I put everything together you can see and what I have done here as I've put the function itself, the original one that I'm trying to figure out on top of my surrogate model. And you can see that the original function and the surrogate model they don't exactly line up, but they're not too, too bad. So, somehow one does recapitulate the other, even though there are fewer than 10 evaluation points for this particular function.

Video 4: Exploration vs exploitation [06:00]

In Bayesian optimisation, we already mentioned exploration and exploitation. We might like to maybe explore the space quite a bit, or we might like to say we narrow in on one particular area and we study that in detail. But how can we represent that mathematically? First off, remember from our previous video that we've got this mean function and we've got an uncertainty function. So, the mean function is where we think that the function might be based on the data that we have evaluated. The uncertainty function is somehow our knowledge that we don't really know what we're talking about.

So, in particular, what are the things that you notice about this mean plus uncertainty function, is that I am more uncertain about the function farther away from points that have been previously evaluated. So, near points that have been previously evaluated, I'm pretty certain that I know what the function looks like. In particular, the way that I'm going to balance exploration versus exploitation is in a so-called acquisition function. So, what I'm showing here are two graphs. On the top, I have both the original function that I wish I knew, and I have the a surrogate function that I am learning over time.

The original function has some uncertainty just because there's some noise associated with the function, it's that I evaluate the function and I get a noisy output back. The surrogate model has uncertainty because that's the nice feature that we want as part of this bayesian optimisation. Now, I might like to evaluate the original function many times in places where the surrogate seems to think that things are going good. Or I might like to evaluate the original function in places where the surrogate model is very very uncertain. So, what you can see on the bottom is a so-called acquisition function that mathematically captures these two goals, that I both want to explore the space and that I want to exploit points that are close in value to the ones that I already think are good. So, what you'll notice in this acquisition function that's on the bottom, is that it has fairly high values in areas where the surrogate model is very uncertain about the original function values. It also has very high values in areas where the surrogate model thinks that we have something that might be a optimum point, in this case we're looking at minimisation.

So, what I'm going to do is that I reduce my problem too. Well, I don't know whether I explore the space or I exploit the space to just one function, this is the acquisition function, and then I optimise only this acquisition function. In this case, I would evaluate this original model now at the point where the acquisition function takes its maximum value, okay? Now, I have evaluated a point. I got a new piece of information away from me. But of course, there's no rest for the weary, right? So, now, I have a new surrogate model that again somehow recapitulates the original function value. And again, I would like to evaluate the surrogate model.

And again, I would like to evaluate the original function value at a point that somehow trades off exploration and exploitation. That is again going to be captured in this new acquisition function. You'll notice that the acquisition function has changed from the previous slide because I got updated information. The points where I am more certain about the function values now are places where I don't want to evaluate the original model anymore. Rather, I want to evaluate the original model in this particular example, in areas where I really just don't know what the function value is like.

We got another point. We updated our surrogate model. We update our acquisition function, and again we maximise our acquisition function to again decide where to evaluate next. This process continues. At this stage, you'll notice that the surrogate function and the original mathematical model are somehow very, very close to one another. That's great. That's what we want to be happening over time. And you'll notice that the acquisition function that's deciding where to evaluate next is getting more and more pointy. The reason it's getting more and more pointy is that basically the bayesian optimisation algorithm has figured out we switch away from exploration, we move towards exploitation, and here is a place where we think we're very close to a global minimum.

Now, this is the mathematical equation that I'm going to associate with this idea. There are a number of different acquisition functions, and acquisition functions are an active area of research. But one acquisition function that's very commonly used is the so-called upper confidence bound. And here what I want to do is I want to maximise the mean of the surrogate function, and then I'm going to have some sort of parameter times the uncertainty associated with the surrogate function. This is a really commonly used acquisition function in bayesian optimisation.

Video 5: When to stop [03:21]

In Bayesian Optimisation, a question that we have to ask ourselves a lot is; okay, when do we stop this, right? Because what I've shown already is that we have this acquisition function that we maximise. This acquisition function trades-off exploration and exploitation. It always tells us where is the next point to evaluate. That's great if I know where the next point to evaluate is. But another thing that I want to know is okay; so, when do I stop? What I'm going to show here is I'm going to show how sort of the diminishing returns works.

I don't have a mathematical rule that says when to stop, because indeed, I don't have convergence guarantees for this particular type of method, except for it to say just keep evaluating and evaluating. Basically my guarantees on Bayesian Optimisation are if you just keep evaluating for all eternity, you will eventually find the global solution to your optimisation problem. That's often annoying, because, for example, my tissue engineering colleagues; they can only do one experiment per month. They do not want to be doing experiments for the rest of their lifetimes, right?

So, when are we going to stop?

Well, a point that you'll often see in Basin Optimisation is that we end up with diminishing returns, and that's what I'm showing here. So, here are a whole bunch of test functions that are very common in Basin Optimisation. They're sort of functions that are interesting and that people have looked at before. And then I'm comparing a number of different Bayesian Optimisation tools against one another. My favourite tool, of course is Entmoot. This is tool that is developed by my research team. But anyway, what's happening here is that we are giving all of the tools the first 50 data points to be exactly the same.

Basically, they have 50 data points to start off with to then start making decisions. If we continue evaluating and continue evaluating, you'll notice that overtime we end up with these diminishing returns. What's going on is that we can't prove it in a mathematical way where we write down some proof that says this is the global optimum. But probably what has happened with these functions is that they really have reached what is effectively the global solution to this problem. So, one of the things you'll see is that they improve quickly and then they may be flatline a bit.

You will also notice in these examples, however, that some improve quickly in flatline. Some seem to just flatline from the very beginning. That would be the kind of example of a surrogate model that you would not want to use for those particular classes of functions. So, I mentioned earlier on that we have sort of different surrogate models that may be more or less appropriate for different classes of functions. You can be sure that the ones that are just sort of never improving on this particular graph are the ones that you don't want to use.

Video 6: Capstone check-in [06:18]

In the previous module, we talked about Bayesian Optimisation. Bayesian Optimisation is critical for the Capstone project, it is indeed what we're going to be doing. Don't stop paying attention to the rest of the course because you can definitely integrate bits from the rest of the course into your Capstone project. But now you have all the tools you need to actually get started on this project. So, the Capstone Project that is a black-box optimisation challenge that came up in NeurIPS 2020. The official competition is over, but I'm really excited about sharing this one with you all because for one, this kind of competition occurs regularly.

I looked at sort of some of the historical competitions that are similar to this, and I cannot find a year since 2009 when there wasn't at least one competition that was very very similar to this one that was available to people. The other thing about this kind of competition is that the problem that is underlying this competition appears very often in machine learning. We're going to be tuning hyperparameters for machine learning models, and so, it's nice to just have a code that you have that does this. How is this competition won? Well, in the NeurIPS 2020 competition, this particular competition was taken over by the really big teams.

The top teams that sort of won the competition: number one was a team from Huawei, number two was a team from NVidia, and number three was a research team from JetBrains Research, this is a big research team in Russia. So, this was taken over by really big researchers and really big teams of researchers. We've talked earlier in the course about when you may like to enter a competition. This is something to think about because, of course, you have limited time. This particular competition would not have been something that you would want to enter if you are trying to look to win a competition or two.

And you would have known that very early on because you would have seen right at the beginning 'Huawei' up at the top and indeed I did. So, I was competing in this particular competition right at the beginning I could see 'Huawei', I could see JetBrains research, and it was like, "Oh look, the really big teams are playing in this particular sandbox". However, I kept competing because I thought it was fun and because I thought I would learn something. So, a relevant question to ask is, how did these particular teams win? And in particular, they won by really knowing their machine learning by really knowing all of the material in this course.

The first-place place team from Huawei, they used Gaussian processes. This is the material from module 12. And then they made a few relatively small modifications. The small modifications that they made, is one that they made modifications to manage the heteroscedastic nature of the functions that are the black box functions that are underlying many important models. So, in particular, this is sort of understanding that a regular normal distribution really doesn't do a good job of capturing the uncertainty. The other modification that the Huawei team made is to deal with the non-stationarity of functions.

This is to say that a length scale over here of how too fast the function is going to change, actually might not be similar to a length scale of a function in a different part of the domain. These modifications are the things that helped them win the competition. The second place team was Nvidia. Remember that Nvidia makes GPUs. They are a device manufacturer. So, they have this cattle grandmaster team: People who are really good at computational machine learning, who are testing out their chips. So, for this team, it was all about using an ensemble algorithm that is, methods that are developed by somebody else, and then distributing the computational complex steps to GPU.

The final team that I'll mention, the third-place team from JetBrains Research. They really studied up on all the material In this particular course. Their third-place strategy was a combination of Gaussian processes, we covered that in module 12, Support vector machines, we cover that in module 14, and nearest neighbour models, we cover that in module eight. So, this is, of course, a closed competition in that the NeurIPS competition is over. But it's an open competition in the sense that all of you can compete against one another at this point. So basically, the competitors in this particular competition are the participants of this course. So, who's going to win? I've thought of sort of three different personas.

Of course, I look forward to seeing what other people do and sort of the personas that you all come up with. One thing you could do is you could be a lazy person. You could clone the winning code base. Take one of the code bases from Huawei, from Nvidia, from JETBRAINS, they're available online. Clone them, submit them. That would be a great way of doing pretty well. But then you know you're not going to do any better than your peers. So, what else could you do? You could be an algorithm developer. This would be similar to the approach that won the 2020 NeurIPS competition, in the sense that you could sort of develop something brand new and sort of show that it improved things.

Another way of thinking about winning would be to go along the lines of the Nvidia team to be a software hacker, to basically take codes that are known to be very very good, and to somehow compete them against one another, decide which of the codes to use, and then make a hybrid. This would also be similar to the so-called Netflix Prize, the way the team won the Netflix Prize