Machine Learning: Executive Briefing

by Simon Allardice

Start Course

Bookmark Add to Channel Download Course

Introduction: Solving New Kinds of Problems

How Anyone Can Write an Introduction to Machine Learning

Let's begin this way. I know you've heard this term, Machine Learning. Let's face it, it's almost impossible to avoid it these days. Machine Learning is transforming the business world! Everyone's talking about Machine Learning! Machine Learning: the next big thing in technology! And you may have even looked at other introductions to it or overviews of machine learning, and most of them take a very similar and predictable approach to this. It's almost like there's a template for how you're supposed to do it. In fact, I'll bet many of you have a sneaking suspicion anyone could write a typical entry-level introduction to machine learning article without actually knowing anything about it, and here's how. First, you would pick your style, your approach; are you going to go with enthusiastic hype or critical hand-wringing? A hype-style introduction to machine learning would go something like this: Machine learning, something-something, self-driving cars, blah-blah, facial recognition of criminals, yada-yada, computers can now beat the best chess players in the world, something-something; identification of twos. Write a few sentences to fill in the blanks, find a stock image of a brain with some glowing techno artwork around it, and there's your introduction, and you can find a bunch of articles of this and be

forgiven for thinking that machine learning is a technology that only applies to cars, chess, criminals, or cancer. And okay, I exaggerate, but only a little, because you do see the same clichés again and again, and it's problem, because if these things aren't what you do, it's very easy to unconsciously internalize a reaction of, yeah, okay, machine learning may be kind of interesting, but it doesn't really apply to me. But it does; it does apply to you, and it will apply to you more and more. Your choice is now, are you going to make use of this, can you learn to leverage machine learning and use it, or just have it be something that's done to you, to your organization. Now the hype-style examples aren't wrong; they are all true. They are great examples of machine learning in action, but they can be a distraction, and focusing on these specific implementations can get in the way of us learning this, of connecting with it, starting to think about machine learning in the right way. So a lot of this short course will be about thinking, thinking the right way, and this is thinking less like a computer programmer and more like a human being, and I'll point out that thinking like a human being is something you can do already so we're off to a good start, but taking this hype approach to machine learning is not the only thing that can get in the way. Let's back up a moment.

Thinking Clearly About Machine Learning

You see, instead of a hype approach, we could've chosen to write a hand-wringing style of machine learning introduction, the excessive displays of concern you see everywhere. Up next, the dark side of machine learning. How dangerous is machine learning? Is machine learning a benefit or a hazard to society? Oh, let me clutch my pearls for a moment! You see, any article that opens this way, you know where it's going: something-something, facial recognition is creepy so it's wrong; genetic profiling will prevent everyone from having healthcare; self-driving cars? Look! One of them crashed; they should all be banned. Machine learning will take all the jobs. In conclusion, Terminator, Sky Net, somebody think of the children! Okay, I don't have a problem with anyone who's critical or skeptical about this stuff. I want you to be critical and skeptical; that's fine, and there are ethical concerns. The problem is when we fall into this trap of either or, where everything becomes black or white, machine learning is either good or it's bad. It deservers either hype or hand-wringing, it's either the most important thing ever, or it's completely over exaggerated and overblown, it's either wonderful or it's creepy; it's either going to help us reach new heights of creative work, or it's going to take all the jobs. Forget all that! We have to avoid these simplistic divisions, these easy answers where everything becomes black or white, true or false, yes or no. First, because this is lazy thinking, and it's simply not true. Machine learning can be, and often is, both kind of amazing and kind of strange and creepy at the same time, but more

importantly why we need to get away from these over-simplistic things are true or false, black and white, yes or no, is because if you want to understand machine learning, to have it click, to have the a-ha moments and the mental light bulbs going off, where it actually makes some kind of sense, then before anything else, before worrying about what exact skills do I need and what programming language should I use, and do I have to be a mathematics nerd, and how exactly will this be useful for me right now, first just get comfortable with one basic idea, that we are now moving away from conventional computer systems that will always give us a simple, well-defined, straightforward yes or no, true or false answer, or an exact amount, and into a much more nuanced world of probabilities and predictions, because this is what machine learning does. In a very fundamental sense, machine learning will let us make predictions based on what's happened in the past; predictions not exact answers. So instead of asking, is something true or is something false, we can ask how probable is it, and have the answer be anywhere on a scale ranging from impossible on one end to certain on the other end, and anything in-between including eh! You see, if you have a business situation where you can provide the rules to always get you to a very straightforward yes or no, true or false, right or wrong answer, or to calculate an exact amount or specific number based on a bunch of conditions, then we don't need machine learning. We don't need machine learning if you can describe all the rules, all the classic conditions. In those situations, conventional computer programming is fine.

Comparing "Conventional" Programming and Machine Learning

Conventional computer programming is great in situations like, if the account balance is less than 0, then display it in a big red font, done. If seat C3 is already occupied on this flight then don't allow seat C3 to be selected by another passenger. If this incoming email is from an address on my VIP contacts list, then put it at the top of my inbox, or we could give a number other rules. If the balance is greater than 5000, and the last withdrawal date was more than 60 days ago, then get the current interest rate, calculate the interest, add them together, save the new balance, log the transaction to the database. We don't need machine learning for any of this kind of stuff, and this leads me to a key point, and it's a point I want to stress for anyone who might have tried to learn about this in the past where it just didn't seem to click, where you might have had the thought, I don't get it, or I kind of get it, I just don't see where I'd use it, and that's not an unusual response even if people don't like to admit it, because if you have a substantial business or technical background in any role; developer, product manager, technical leader, whatever it is, perhaps even decades of experience, then you may find yourself at a slight disadvantage when learning this. A slight disadvantage. Here's why and here's what to do about it. You see, with years

or decades or experience, you get comfortable jumping from different technologies, new programming languages, new environments, new models like going from desktop projects to web projects to mobile projects, and you've probably learned that you can shortcut your learning of a new subject by mentally bringing over a problem that you're already familiar with. I know how to fix this using technology A so let me take this problem to technology B. I'll take a problem I've solved on the desktop or the web, and I'll see how to take this problem to the cloud. I'll take a project I know how to implement in .NET and I'll figure out how to do it in Node, but that approach, and you may not even be consciously aware you're doing it, that approach will backfire, because machine learning is not a new way to solve the kinds of problems you're already familiar with. Let me say that again. Machine learning is not a new way to solve the kinds of problems you're already familiar with. Now look at machine learning as a way to solve new kinds of problems. Okay, that's a little cryptic, and when I say new kinds of problems, I don't mean problems you have to invent or things you didn't even realize were problems, or entirely new categories in the problem space. No, more often these are situations that you're already perfectly aware of. You might not even call them problems. You might think of them as business decisions or issues or tasks, but where in the past you might've assumed that computers just couldn't help you or couldn't help you that much. Like what?

Embracing Everyday Machine Learning Examples

I want to list off a few machine-learning examples, but I want to avoid those epic self-driving cars, robots, and satellites, and let me go from the bottom up with a few simple intentionally mundane, intentionally unexciting examples like, what's the best price to sell our new product? How often should we send marketing emails? Should we segment those emails by category, and if so, what would those categories be? How likely is it that this particular customer will renew their subscription, and is there anything we could do that would keep them here? When people are talking about us online, is the conversation good or bad, and does it even matter? And perhaps, right now, out of thousands of people using our website, there are two people making really strange requests to the API; should somebody be informed about that? Now you might deal with these situations already where in a lot of companies, you might expect to begin with a little bit of data, but where that information is just input for a purely human process, a human decision. So there are plenty of companies where if you asked, do you have a back-end application to handle a customer making a purchase of a product, they would say, well sure, of course we have a back-end system to do that. But if you ask those same companies, do you have an application to handle how to price that new product, they would say, well, no, we bring up some spreadsheets,

we look at some history, and we sit around a table, and we argue about what it means. Or perhaps those companies would say that sure, of course they have a back-end application to send their marketing emails, but when it comes to deciding how often those are sent and how segmented those email campaigns are, well, we think that's up to Alice and Bob in Marketing; they decide what happens and they figure it out, or we know people are having conversations about us online, but it's the job of the social media manager to read every Tweet and Facebook comment and figure out whether it's good or bad. And these are just a few simple, every-day, and again intentionally mundane examples of things machine learning could actually help you with it. And it's not that you're judgment or your intuition or your gut or your experience goes away; no, it's still important, but we can make it a little more data driven, a little more objective. So we need to build our own ability to recognize the situations, to widen our beliefs about what computer systems can and can't help us with to start to recognize the places where we might have previously thought, okay, a computer can do that thing, but a human being does this thing. And what does help when you're getting your head around machine learning is to recognize that the classic examples of machine learning that you may never personally need, and you use them to improve your own ability to recognize those situations. Things like whether an email is spam or not spam, or if a purchase is fraudulent or not fraudulent. How can we improve online recommendations, or how could we recognize images that contain a picture of a flower? And it's easy to think well, we'd use Gmail to figure out the spam thing, and our credit card processor will tell us if something is fraudulent or not, and well, we don't have a customer base like Amazon or Netflix so we don't need a recommendation system, or well, I don't even know where to begin how to tell you how much we don't care about image recognition. So none of this applies, give me other examples. Now I get it, but you see, those simple examples can be excellent to work with to help you improve your knowledge, as long as you take this up a notch with the concepts, because they're simple to describe, but we can use them to talk about probability and ideas like accuracy and loss and how to choose different machine learning algorithms to solve different kinds of problems, and there are just a few things, just a very few things that machine learning is excellent at, done well scarily good, and plenty of things that it isn't. And we're going to talk about the differences, but as we start to identify these different situations, well then what? I mean, what is machine learning really? How does it apply? How can we actually do this? Well, here we go!

But What Is Machine Learning, Really?

Comparing Multiple Definitions of Machine Learning

If you've ever done a web search on the phrase what is machine learning, you'll quickly discover there is no one single accepted definition, there's no one phrase to rule them all. You'll find a lot of vague and all-encompassing abstract definitions, like this one from Carnegie Mellon University. Machine Learning asks, "How can we build computer systems that automatically improve with experience, " okay, "and what are the fundamental laws that govern all learning processes? " This sounds impressive, but it's not exactly something you can ask a team to do on a Monday morning. Okay, today, Fred, you're going to change the color of the font on the Submit button, and, Jan, how about you get busy on the fundamental laws that govern all learning processes. Or, you might find definitions that are so specific that they will make your eyes glaze over. There's a widely quoted definition of machine learning you'll find in a lot of textbooks, that machine learning is, "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E. " Okay, I don't find this exactly a lot of help in how to get started with all of this. So, to start our journey, let us deal with this term, machine learning, and we naturally deal with quite a few others. Because as soon as you take one step into this world, you should expect to be immediately surrounded with more phrases, more jargon, not just machine learning, but also deep learning, supervised learning, unsupervised learning, reinforcement learning. And there are other words and phrases that come along for the ride, like data science, data mining, and artificial intelligence. If you read a little deeper, you will find incredible amounts jargon, depth-wise separable convolutions and multivariate adaptive regression splines. In an introduction, we can't deal with all of these, but we will get to the important ones. So let's clear one question up. We're talking about machine learning, a computer system that learns. Well, so what is the difference between that and artificial intelligence?

The Difference between AI and ML

Artificial intelligence is sometimes a challenging term to deal with. It brings a lot of baggage with it. It's been around for so long, it's almost retro. I feel like I should be saying, with artificial intelligence, we can now take the information superhighway to the new cyberspace frontier. But

okay, retro or not, let's be clear. Machine learning is a kind of artificial intelligence. If we take AI as broadly as possible, as, well, anything we can make a computer do that's kind of smart, machine learning is one way to make that happen. It's not the only way, but it's where we try and make the computer smart by providing examples instead of trying to make the computer smart by explicitly programming a bunch of rules into it. Now that still might sound a little vague, so let me give you a conceptual example which will involve mime and audience participation. If I want to play poker with you and you don't know how to play, I'm going to explain the rules to you. I will first need to figure out your base knowledge. Maybe you've never played card games at all, so I'd first have to describe the deck and the different suits and the order of play and what makes a straight or a flush or a full house and what beats what. Now it's possible that if you just sat down and watched people play poker, you would eventually figure out the rules, but that would take a long time. This is a situation where we would both find it easier to begin with me explaining the explicit rules. And just knowing the rules won't make you a good poker player, but we would start there. If I could then provide you with more rules, more explicit situations of how to handle edge cases, you would, hopefully, get better. But in the same way, if I wanted to write a computer program to play poker, I would need to program the rules of poker into it. Now just knowing the rules won't make it a good computer program, but we would begin there. But, let's take a different kind of problem. If I wanted you to help me organize a stack of photographs and to identify any photographs of, well, how about I not begin by explaining the rules. How about if I just show you positive examples and negative examples. We wouldn't even have to speak the same language. So, let's try this, let me explain what I want with the power of mime. First. (Cheery music) Now you tell me. If I show you a new pieces of information, you tell me, thumbs up or down? And? Congratulations, you have now personally done one of the things that machine learning is very, very good at, classification. Provide enough positive and negative examples and you can classify new input as falling into one category or another. The way you learned what I wanted, by me providing you with examples, with data, not by me explaining what the explicit rules were. I didn't begin by saying, I need you to identify Equus caballus, a solid-hoofed, plant-eating, domesticated mammal with a flowing mane and tail. No, your level of understanding of a horse was irrelevant. See, I didn't want to have to explain that a horse has four legs, but sometimes in a photograph you'll only see two, and what the different colors could be. And sometimes, you won't see any legs. And sometimes it'll be one horse and sometimes several. And sometimes the horse will have a rider and sometimes it won't. But Simon, you might say, I already knew what a horse was, so you're using my understanding of it. A valid point, but let's say that instead of pictures of horses, I had shown you this. You see, if I show you enough, you don't actually have to know or understand what it is you're looking at, whether this was a fruit or

vegetable, whether it's delicious or poisonous, whether it's large or small or even a prop from a movie, you can just recognize the characteristics of one of those things. And this is one of the problems that machine learning is very, very good at, as long as we can provide enough high-quality examples, but what does count as one of these things and what does not count as one of these things? But let me be clear, classification is not just about images. We can take the same concept to things like is an email spam or not spam? Does recent customer activity on a website seem to match typical characteristics of high-value customer or a low-value customer? And classification does itself get more specialized, we'll talk about that in just a moment, but it is an ideal micro-example of general machine learning ideas. So, for the next few minutes, we're going to go with our own working definition of what is machine learning?

Writing a More Useful Working Definition of Machine Learning

So let's begin with this following definition of machine learning, and imagine I write it on a chalkboard so we can tweak it and refine it over the next few minutes. Machine learning allows it to take existing data, analyze it to identify patterns, and use the results to make better predictions about new data. Okay, a bit more useful, but it's still a little vague. It still begs a lot of questions. What kind of existing data? How much data? And who analyzes it, us or the computer, and how? And if we identify patterns, well, what kind of patterns? Who says what's a pattern and what isn't a pattern? And predictions, well, what does a prediction look like? And how? How do you get started with all of this? Well, we'll get to all of this. But I wrote this definition in this way because when using machine learning, there's a series of specific steps we'll go through. There is a sequence to this, and it's a different sequence than you would do in either conventional computer programming or even just standard data analysis. So, to get comfortable with this sequence, I'm going to cycle around this definition a couple of times, line by line. We'll make a few changes and go deeper each time around.

Training a Machine Learning Model

Step 1: Beginning with Existing Data

Let's take this first line. You see, machine learning requires us to have existing data. Now before you say, well duh, Simon, all computer programs need data. They take data in, they spit data out. Well I'm not talking about the data our application will use when we run it, that's in the future. That comes later. I'm talking about right at the beginning of the development process, we need data purely to learn from. And we want real data, and hopefully, a lot of real data. The more data, the better. The more examples I can provide, the better the computer should be able to learn. And if you ask, what kind of data? Well, what do you have? Sales orders, customer transactions, activity on your website? Do you have images, do you have emails, do you have the last million phrases that people have typed into your search engine? You see, while we can work with many different kinds of data, it doesn't mean we just take everything that we have in our databases and our file systems and just dump it all in. No, our data will need to be prepared. We need it to be high quality. We will filter it down to only the data we're interested in right now, and we have to tidy that data up. We have to get rid of garbage entries or missing pieces of information. Get rid of data that's ambiguous or confusing. We're trying to train the computer to understand the difference between what we want and what we don't want. So our data also needs to be labeled. If we were doing a machine learning classification task, like I showed a moment ago, we would need to categorize that initial data into positive examples, horse, and negative examples, not a horse. But let me be clear. Preparing the data is a major, major, major piece of machine learning. This is a step that takes significant time, thought, and attention. But let's move on, because having data is not an unusual step in any programming, whether we're using machine learning or not. But the next step is different.

Step 2: Analyzing Data to Identify Patterns

So we begin with data, organized, cleaned, and prepared data. We then analyze it to identify patterns. Well, what does that mean? Who analyzes it, us or the computer? Now in conventional software development, we might still begin with some existing data, perhaps databases or even spreadsheets that we have, but where we'd have our analysts and our developers and business owners and stakeholders all sitting around the table and looking at it, and trying to figure out what that data means and what we're going to do with it, and after much discussion and argument and whiteboarding and flow charting, we could then send off all the programmers and they'd go and write all the business rules and coding to handle all those different situations, hopefully. If you've done this, anything from looking at the accounts of a small business to looking at analytics for a website, or looking at historical transaction information, you know it's incredibly easy to focus on the wrong thing. And the more data you have, the more difficult it gets. So

what's different with machine learning is instead of just us, as human beings, looking at the data and trying to figure out what's important about it, and we still do that, but we will also apply a machine learning algorithm to our data, sometimes called a learning algorithm, or even just a learner. Now if you're new to this, you might ask, so, do the programmers in my organization need to write these machine learning algorithms? No. The vast majority, the overwhelmingly vast majority of companies and individuals do not need to write and never need to write machine learning algorithms. Instead, we'll take a machine learning algorithm that somebody else has already written and we will apply it to our data. But Simon, you say, you're telling me we can just pick some machine learning algorithm off the shelf and just feed our data into it? Well, a bit oversimplified, but kind of, yes. Let me talk about that.

Exploring Machine Learning Platforms and Frameworks

When you want to create a database, you choose a database management system. If you want to make a website, you choose a web server or a hosted platform. And in the same way, if you want to create an application that uses machine learning, you're going to use an existing machine learning platform, or a machine learning framework. And there's a lot of them, both commercial and open source. There are hosted options from Microsoft, Google, Amazon, and IBM. There are open source frameworks like TensorFlow, Torch, and Caffe. Now sure, these aren't identical, they each have their own strengths and downsides, but for what we're talking about here, the differences aren't important because they all include multiple machine learning algorithms that we can pick and choose from, depending on what kind of task we're trying to do. But I don't want to suggest it's as simple as just clicking a button. No, this is another part where you can't remove the human because even though we may not write these machine learning algorithms, there's still a lot of work to do in choosing the correct algorithm, applying it and configure it and testing it. And this is another part that takes time, attention, and most importantly, understanding what it is we're actually trying to accomplish with our data. Now if you have a classic team of people working on machine learning projects, this would be one of the responsibilities of the data scientist role. And most will tell you it is not a single task, it is a process where you'll often experiment with several different machine learning algorithms, testing them to see which ones are giving you the best results. We are not yet at the point where we could take every scrap of data in our organization, feed it all into a machine learning hopper, and just stand back to wait for it to spit out lines of wisdom, like, you should offer your products in blue. Send your marketing emails on Monday morning. And oh, did you know your VP of Operations is about to quit? No. Before we can start doing any of this and choosing and applying a machine learning algorithm, we still have

to decide exactly what it is we're looking for right now. Let's say someone handed me a flash drive with a 100 GB of unexciting website activity data. I could take that data to help identify fraudulent purchases, or I might use the same data to identify customers with high potential value. Or, use the same data to identify hacking attempts. These are all valid possibilities. And the best machine learning algorithm would be different for each situation. But when we successfully apply a machine learning algorithm to analyze our data, to learn from it, the result we get from doing that is called a model. We say that we have trained a model. And the model is a file, we could save it to our desktop. We could copy it from machine to machine. It is generated code, but not in the sense of here's some generated code I could give to my programmers and they could start editing it and using it as a starting point for further development. No. The trained model is not a starting point. The model is the end result of our machine learning process. It's what we're going to take and use in the next step.

Making Predictions with a Trained Model

So when we've trained our model, this result of applying a machine learning algorithm to our data, there's a bunch of different ways we can use it. We could take that model and import it into a software application we're building. And there are different frameworks we can use to deploy a trained model into, say, an iOS application or into a web back end. We can take the same model and upload it and host it in a cloud service, like Microsoft's Azure, Amazon's AWS, Google Cloud, IBM Cloud. We could give it an address, give it a URL, and expose it as an API that we could call from other applications. Pass in some new data, get a result back. We can also use some models in business applications like Excel and Tableau. But however we decide to use it, we have trained our model with existing data, and we can now pass in new data and get results, get predictions. Not perfect predictions, probably, but better predictions. There's a couple of things worth pointing out. When you do a classic classification task, the result you get is on a probability scale. On a scale of 0 to 1, the model says this email is a 0.82, and that email is a 0.49. Now, what you do with those numbers is completely up to you. But predictions aren't always just on probability scale. We can also get an amount, like predicting a price. The model says a good price to sell this new house based on the information you provided is $372, 512. When you use a trained model and you get a result, an answer from it, what you don't get is an explanation. This is sometimes called the black box of machine learning. We feed a bunch of new data into the model, we get a result, we don't get an explanation of why that result happened. And you might be puzzled by that result. The problem is, if you want more visibility into the decision that the model is making, you might need to go deeper into the algorithms, and it's time we did that.

The Marketplace of Machine Learning

Understanding ML Classification Algorithms

Machine learning is good at only a few things. Some machine learning algorithms fall into just a few kinds of task, a few categories. First is the one we've seen already. Classification. Does this thing fall into category A or category B? But, classification itself has two main styles. One is where you are just looking for whether it's A or B. Spam or not spam. Horse or not horse. You want just two options. This or that. And this is called two-class or binary classification. But we also have situations where we have multiple categories, not just horse or something else, but is it a horse or is it a car or a person or a mountain top or a hoverboard or whatever. And this would be multi-class classification. When we make the distinction, and it is just between whether it's 2 kinds of things or more than 2, 3, 4, 1000, we make the distinction because there are some machine learning algorithms that are better at two class or binary classification and some algorithms that are more suited for multi-class classification. And classification as we've seen can have a huge amount of tasks just beyond image recognition. Things like spam filtering is classification. Or, determining customer behavior is high value or not. That can be classification as well. This is not the only thing machine learning is good at. We also have a category called regression.

Exploring Regression in Machine Learning

Regression is what we use when the prediction we want is an actual number, an amount, like what price should we buy or sell this stock at? Or, how many subscriptions do we think we will sell in the next quarter? Or, how long will it take to complete this project? And where we know there are large numbers of contributing factors where we just can't expect to actually write the rules to determine this. Think of the innocent question, I want to put my house on the market, how much should I ask? Well, we know there's a bunch of factors that play into this. The location has multiple parts to it. Postal code, neighborhood, street, north or south exposure. Are there views? How private is it? What about size, or age, or condition, or quality? Is it empty, is it staged? How many stories does it have? What's the lot size? Is there parking? Is there a garage? How's the stock market doing right now? What level of buyer do you need? What time of year is it? There are dozens, possibly hundreds of factors to take into account. And it is immensely difficult, almost

impossible to come up with an exact set of rules for how these all affect each other. But still, it's not random. So, with enough historical data, with enough factors, we should be able to make some kind of prediction of a sale price, even with a margin of error, and we would use regression for this and for things like this. But it would be our job to take the raw data, in this case house sales, and prepare it just as with providing examples of what we want in classification. When we're doing regression and using a regression algorithm, we tell it what number we're trying to predict, like house price, and all the other data items we would like this algorithm to take into account. Because when we have situations with lots of different pieces of data, it's often incredibly difficult for us as human beings to recognize the tiny subtle distinctions between which ones are actually more important than other ones. But a machine learning algorithm that can look at massive amounts of data and very objectively help us understand the effect that it really does have on the end result.

Comparing Supervised and Unsupervised Learning Algorithms

Both regression and classification rely on us to provide a lot of instruction. We provide a lot of intentional data and direction for how the computer should learn. This is good, this is bad. This piece of data is more important than that piece of data. And we refer to these not just as machine learning, but supervised learning. Supervised learning is the most common kind of machine learning application. Many, many tasks can be accomplished using regression or classification. But it begs the question, if that's supervised learning, then what else is there? Well, there is also unsupervised learning where we don't provide the same amount of direction. We don't label all our data into categories or we don't tell the algorithm that we want it to come up with this number based on those other numbers. Two examples of unsupervised learning are clustering and anomaly detection. These are really useful, but they're not as widely applicable as classification and regression. So first, clustering. This is where we can provide a set of data and we want the algorithm itself to tell us what are the groupings, the categories that exist inside of it. An anomaly detection is a bit like classification, but it's one we use in a situation where most of our data is normal. We have typical data. We can tell it what looks normal, but where we don't want to explicitly tell the computer what looks like not normal. Anomaly detection is used in things like network traffic analysis where you're not looking for a particular style of network attack, you're just looking for anything out of the ordinary. Or say, heartrate. We have a lot of normal heartrate data and then we have not normal, something weird is going on. So, let me recap and summarize the main situations that machine learning's really good at. Classification. I need to know what this thing is. There's regression. I need to know the value of this. There's

clustering. Help me understand the structure of all of this data. And there's anomaly detection or, huh, that's weird.

Reviewing the Process: Where to Go from Here

So let's revisit our working definition. We touched this idea of preparing the data. We need to clean it, we need to get rid of garbage entries. We need to label our data and provide information about either the categories we're looking for, or selecting the data we think is important to bring in to our regression analysis. Then we take this data and we would split it into a training set and a testing set. It's usually about an 80-20 split. We take 80% of the data and use it to train the model and then the other 20% to validate and test the model's predictions. We would then choose a learning algorithm, and again, we expect this to be an iterative process. We would try one algorithm, run some tests, try another algorithm and run those tests and see if the results are better or not. It's easy when you first get started to assume that these first two steps are what we need to get passed to get to the interesting stuff. But in machine learning, this is the work. These first two steps are the most important, the most time-consuming part of it all. Because once you have a trained and tested model, it's all downhill from there. You just use it. Once you have a trained model, you have a variety of options for how to deploy it, how to use that model. And if we think we have better training data a little bit down the line, we could train the model again. If we get better results from that new model, we could redeploy it. But again, there are excellent machine learning frameworks, things like TensorFlow and Torch. There are commercial hosting options. You have Azure Machine Learning, AWS Machine Learning, Google Cloud, IBM Watson, and you'll find there's a lot of common ground between them. But the thing is, right now it really doesn't matter that much. You're not trying to be an expert on any one of these frameworks, you're just trying to get to grips with the basics of machine learning itself. But if you are looking to experiment, to get started with this, these days I would suggest you often begin with one of the web-based tools to ease your learning curve. For example, Microsoft Azure Machine Learning Studio. This is a visual drag and drop environment for importing and preparing your data, for picking machine learning algorithms, and for training and testing your model. But there's also Amazon SageMaker, Google Cloud's Datalab, and so on. And when using things like this, you'll see that you can choose from multiple different algorithms. You have different options to select from the categories we've talked about like anomaly detection, classification, clustering, and regression. And you'll find many options within each of those categories. You don't need to understand all these different algorithms going into them. Some of them you'll never need to understand. You'll find that some algorithms are faster to train. That might be an issue if you need

to retrain your model regularly with new data. Some other algorithms are more configurable. Some algorithms allow more visibility into the decision process. And with some, you'll just say, well, we tried this one but we got better results with that one. We're not entirely sure why, but we're going with that. And in many cases, you will try a bunch of different algorithms and test your results. And as I said early on, I highly recommend embracing some of the classic cliched machine learning examples. Even if you knew you were never going to use machine learning image recognition, it's a good thing to experiment with because of the way it makes you think about probability and how to split your data into training and testing sets and getting the right perspective on where most of the work is actually done in a machine learning project, because that's also the attitude to have when you're trying to find ways to apply it in your own life. You don't need to always go with these epic self-driving cars and facial recognition criminals options, you just need to think about how to make life a little bit easier for your customers. Make life a little bit easier for yourself. Thanks for joining me. See you next time. We hope you enjoyed this course. If you're interested in more content for technical leaders. Content we keep short and focused with the up to date information you need to be informed and make decisions but without getting buried in the details. Find other courses like this at plrsig.ht /exec

Course author

Simon Allardice

Simon is a staff author at Pluralsight. With over three decades of software development experience, he's programmed in every discipline: from finance to transportation, nuclear reactors to game...

Course info

Level	Beginner
Rating	★★★★★ (176)
My rating	★★★★★
Duration	0h 39m
Released	11 Jul 2019

Share course