



Machine Learning & Artificial Intelligence: Crash Course Computer Science #34

Crash Course: Computer Science

<https://youtube.com/watch?v=z-EtmaFJieY>

<https://nerdfighteria.info/v/z-EtmaFJieY>

Hi, I'm Carrie Anne and welcome to Crash Course Computer Science.

As we've touched on many times in this series, computers are incredible at storing, organizing, fetching, and processing huge volumes of data. That's perfect for things like e-commerce websites with millions of items for sale and for storing billions of health records for quick access by doctors.

But what if we want to use computers not just to fetch and display data, but to actually make decisions about data? This is the essence of machine learning, algorithms that give computers the ability to learn from data, and then make predictions and decisions.

Computer programs with this ability are extremely useful in answering questions like 'is this email spam?', 'does a person's heart have arrhythmia?', or 'what video should YouTube recommend after this one?'.

While useful, we probably wouldn't describe these programs as intelligent in the same way we think of human intelligence. So even though the terms are often interchanged, most computer scientists would say that machine learning is a set of techniques that sits inside the even more ambitious goal of artificial intelligence or AI for short.

===== ntro (0:59-1:08) =====

Machine learning and AI algorithms tend to be pretty sophisticated. So rather than wading into the mechanisms of how they work, we're going to focus on what the algorithms do conceptually.

Let's start with a simple example: deciding if a moth is a luna moth or an emperor moth. This decision process is called classification and an algorithm that does it is called a classifier.

Although there are techniques that can use raw data for training like photos and sounds, many algorithms reduce the complexity of real-world objects and phenomena into what are called features. Features are values that usefully characterize the things we wish to classify.

For our moth example, we're going to use two features: wing span and mass. In order to train our machine learning classifier to make good predictions, we're going to need training data. To get that, we'd send an entomologist out into a forest to collect data for both luna and emperor moths.

These experts can recognize different moths. So they not only record the feature values, but also label that data with the actual moth species. This is called labeled data.

Because we only have two features, it's easy to visualize this data in a scatter plot. Here, I've plotted data for 100 emperor moths in red and 100 luna moths in blue. We can see that the species make two groupings, but there's some overlap in the middle.

So it's not entirely obvious how to best separate the two. That's what machine learning algorithms do: find optimal separations.

I'm just going to eyeball it and say that anything less than 45mm in wingspan is likely to be an emperor moth. We can add another division that says additionally, mass must be less than 0.75 in order for our guess to be emperor moth.

These lines that chop up the decision space are called decision boundaries. If we look closely at our data, we can see that 86 emperor moths would correctly end up in the emperor decision

region, but 14 would end up incorrectly in luna moth territory. On the other hand, 82 luna moths would be correct, with 18 falling onto the wrong side.

A table like this, showing where a classifier gets things right or wrong, is called a confusion matrix. Which probably should have also been the title of the last two movies in the *Matrix* trilogy.

Notice that there's no way for us to draw the lines that give us 100% accuracy. If we lower our wingspan decision boundary, we misclassify more emperor moths as lunas. If we raise it, we misclassify more luna moths.

The job of machine learning algorithms at a high level is to maximize correct classifications while minimizing errors. On our training data, we get 168 moths correct and 32 moths wrong. For an average classification accuracy of 84%.

Now using these decision boundaries, if we go out into the forest and encounter an unknown moth, we can measure its features and plot it onto our decision space. This is unlabeled data. Our decision boundaries offer a guess as to what species the moth is. In this case, we'd predict it's a luna moth.

This simple approach of dividing the decision space up into boxes can be represented by what's called a decision tree, which would look like this pictorially or can be written in code using IF statements like this.

A machine learning algorithm that produces decision trees needs to choose what features to divide on. And then for each of those features, what values to use for the division.

Decision trees are just one basic example of a machine learning technique. There are hundreds of algorithms in computer science literature today and more are being published all the time. A few algorithms even use many decision trees working together to make a prediction.

Computer scientists smugly call those forests, because they contain a lot of trees. There are also non-tree-based approaches like support vector machines, which essentially slice up the decision space using arbitrary lines.

And these don't have to be straight lines. They can be polynomials or some other fancy mathematical function. Like before, it's the machine learning algorithm's job to figure out the best lines to provide the most accurate decision boundaries.

So far, my examples have only had two features, which is easy enough for a human to figure out. If we had a third feature, let's say length of antennae, then our 2D lines become 3D planes, creating decision boundaries in three dimensions.

These planes don't have to be straight either. Plus a truly useful classifier would contend with many different moth species. Now I think you'd agree, it's getting too complicated to figure out by hand, but even this is a very basic example: just three features and five moth species. We can still show it in this 3D scatter plot.

Unfortunately, there's no good way to visualize four features at once, or twenty features, let alone hundreds or even thousands of features. But that's what many real-world machine learning problems face.

Can you imagine trying to figure out the equation for a hyperplane rippling through a thousand-dimensional decision space? Probably not, but computers with clever machine learning algorithms can. And they do, all day long, on computers at places like Google,



Machine Learning & Artificial Intelligence: Crash Course Computer Science #34

Crash Course: Computer Science

<https://youtube.com/watch?v=z-EtmaFJieY>

<https://nerdfighteria.info/v/z-EtmaFJieY>

Facebook, Microsoft, and Amazon.

Techniques like decision trees and support vector machines are strongly rooted in the field of statistics. Which has dealt with making confident decisions using data long before computers ever existed. There's a very large class of widely used statistical machine learning techniques.

But there are also some approaches with no origins in statistics. Most notable are artificial neural networks, which were inspired by neurons in our brains. For a primer of biological neurons, check out our 3-part overview here.

But basically, neurons are cells that process and transmit messages using electrical and chemical signals. They take one or more inputs from other cells, process those signals, and then emit their own signal. These form huge, interconnected networks that are able to process complex information, just like your brain watching this YouTube video.

Artificial neurons are very similar. Each takes a series of inputs, combines them, and emits a signal. Rather than being electrical or chemical signals, artificial neurons take numbers in and spit numbers out. They are organized into layers that are connected by links, forming a network of neurons, hence the name.

Let's return to our moth example to see how neural nets can be used for classification. Our first layer, the input layer, provides data from a single moth needing classification. Again, we'll use mass and wingspan.

At the other end, we have an output layer with two neurons, one for emperor moth and another for luna moth. The most excited neuron will be our classification decision. In between, we have a hidden layer that transforms our inputs into outputs and does the hard work of classification.

To see how this is done, let's zoom in to one neuron in the hidden layer. The first thing a neuron does is multiply each of its inputs by a specific weight. Let's say 2.8 for the first input and 0.1 for its second input. Then it sums these weighted inputs together, which in this case is a grand total of 9.74.

The neuron then applies a bias to this result. In other words, it adds or subtracts a fixed value. For example, -6.0 for a new value of 3.74.

These bias and inputs weights are initially set to random values when a neural network is created. Then an algorithm goes in and starts tweaking all those values to train the neural network, using labeled data for training and testing.

This happens over many interactions, gradually improving accuracy. A process very much like human learning.

Finally, neurons have an activation function, also called a transfer function that gets applied to the output, performing a final mathematical modification to the result.

For example, limiting the value from a range from -1 and 1, or setting any negative values to 0. We'll use a linear transfer function that passes the value through unchanged. So 3.74 stays as 3.74.

So, for our example neuron, given the inputs 0.55 and 82, the output would be 3.74. This is just one neuron, but this process of weighting, summing, biasing, and applying an activation function is computed for all neurons in a layer. And the values propagate forward in the network one layer at a time.

In this example, the output neuron with the highest value is our decision: luna moth. Importantly, the hidden layer doesn't have to be just one layer. It can be many layers deep. This is where the term Deep Learning comes from.

Training these more complicated networks takes a lot more computation and data. Despite the fact that neural networks were invented over 50 years ago, deep neural nets have only been practical very recently thanks to powerful processes, but even more so wicked-fast GPUs.

So thank you gamers, for being so demanding about silky-smooth frame rates.

A couple of years ago, Google and Facebook demonstrated deep neural nets that could find faces in photos as well as humans. And humans are really good at this. It was a huge milestone! Now deep neural nets are driving cars, translating human speech, diagnosing medical conditions, and much more.

These algorithms are very sophisticated, but it's less clear if they should be described as intelligent. They can really only do one thing. Like classify moths, find faces, or translate languages.

This type of AI is called weak AI or narrow AI. It's only intelligent at specific tasks. But that doesn't mean it's not useful. I mean, medical devices that can make diagnoses and cars that can drive themselves are amazing.

But do we need those computers to compose music and look up delicious recipes in their free time? Probably not. Although, that would be kind of cool.

Truly general purpose AI, one as smart and well-rounded as human, is called strong AI. No one has demonstrated anything close to human level artificial intelligence yet.

Some argue it's impossible, but many people point to the exposure of digitized knowledge like Wikipedia articles, webpages, and YouTube videos as the perfect kindling for strong AI.

Although you can only watch a maximum of 24 hours of YouTube a day, a computer can watch millions of hours. For example, IBM's Watson can consults and synthesizes information from 200 million pages of content, including the full text of Wikipedia.

While not a strong AI, Watson is pretty smart and it crushed its human competition in Jeopardy way back in 2011. Not only can AIs gobble up huge volumes of information, they can also learn over time. Often much faster than humans.

In 2016, Google debuted AlphaGo, a narrow AI that plays the fiendishly complicated board game: Go. One of the ways it got so good and able to beat the very best human players, was by playing clones of itself millions and millions of times.

It learned what worked and what didn't and along the way discovered successful strategies all by itself. This is called reinforcement learning and it's a super powerful approach. In fact, it's very similar to how humans learn.

People don't just magically acquire the ability to walk. It takes thousands of hours of trial and error to figure it out. Computers are now on the cusp of learning by trial and error. And for many narrow problems, reinforcement learning is already widely used.

What will be interesting to see is if these types of learning techniques can be applied more broadly to create human like strong AIs that learn much like how kids learn, but at super accelerated



Machine Learning & Artificial Intelligence: Crash Course Computer Science #34

Crash Course: Computer Science

<https://youtube.com/watch?v=z-EtmaFJieY>

<https://nerdfighteria.info/v/z-EtmaFJieY>

rates.

If that happens, there are some pretty big changes in store for humanity. A topic we'll revisit later.

Thanks for watching, see you next week!

===== Credits (11:18) =====

Crash Course Computer Science is produced in association with PBS Digital Studios. At their channel, you can check out a playlist of shows like Origin of Everything, Reactions, and the Art Assignment.

This episode was filmed at the Chad and Stacey Emigholtz Studio in Indianapolis and it was made with the help of all these people and our wonderful graphics team, Thought Cafe.

That's where we're going to have to halt and catch fire. I'll see you next week!