What is machine learning?

bossequity.com/news/what-machine-learning

by Adrian Redgers

Conventional Fraud Detection

Machine learning means training a computer to do something by showing it examples rather than by explicitly programming it. Suppose your website accepts transactions for your catalogue of goods and you want to classify a new transaction as likely fraudulent or not.

A conventional solution might be to take all the details of the new transaction and tally up how many of its features match features you know too often occur in fraudulent transactions. If that tally is greater than some threshold you have set, then you could mark the new transaction for manual review by your Fraud Department, or, if less than the threshold, let it pass without review.

You could weight the features that contribute to the tally – For example, a change of delivery address might be twice as strong an indicator of fraud as the fact that the transaction was for jewellery - whilst a returning customer might be an indicator that this is unlikely to be fraud. In which case, you should reduce the tally that you test against the threshold.

To use this method, you need to know the features to look out for, the amounts you want to weight them by and the threshold you want to set. Your Fraud Department may have experience of which features it has seen previously in fraudulent transactions, and a bit of work with a spreadsheet could give you the frequency with which they occur. You can then use this information to set weights and threshold. But this still may not catch all cases.

For example, a returning customer with no other indicators might be a “not fraud” indicator, but a returning customer buying jewellery and changing the delivery address might be a very strong indicator of a hacked account. There may also be some indicators you have missed altogether such as the importance of the time of day of the transaction - maybe online crooks are busy when most genuine customers are at work, asleep - or both!

Learning Algorithm Identification of Fraud

With machine learning, you train a computer by showing it examples of historically fraudulent and non-fraudulent transactions and let the machine learning algorithm discover for itself which features, or combination of them, are important for detecting fraud.

You don't necessarily get to know what the important features of a fraud are, nor their weights, but you do get a machine that can accurately classify a new transaction – exactly like an examiner in your Fraud Department who has amazing hunches but cannot tell you why.

What is the Difference Between Supervised and Unsupervised learning?

This kind of training, where you train the computer on a set of labelled examples, is called "supervised learning". In this case, a set of historical transactions are labelled "fraudulent" or "ok". In "unsupervised learning" - the other main type of training - you have examples but no labels.

In the case of our online transactions you just have a set of transactions without the "fraudulent" or "ok" labels.

Supervised learning is used for classification problems, like our fraudulent/ok classifier, and for creating artificial models of complex systems: for example, predicting the amount of vibration of a gas turbine at different speeds, or the value of Bitcoins tomorrow or finding the best next move in a game of chess.

Unsupervised learning, on the other hand, finds natural classifications or groupings in the data it sees. Running an unsupervised learning algorithm on only the historically fraudulent transactions might show that they seem to be grouped into, say: transactions containing change of delivery address, or for jewellery - which we knew - and it should also find that grouping we mentioned of returning customers buying jewellery and changing delivery address. It might also discover another grouping: refund transactions that take place in the early hours of the morning. Our fraud department would be interested to know these groupings.

Running unsupervised learning on only non-fraudulent transactions might Identify market segmentation groupings, for example, that a significant group of transactions are for exchanging clothes that were gifts, which our marketing department would be keen to know about.

And who knows, running unsupervised learning on all the transactions, without labels, might discover the fraudulent and ok groupings by itself, and then go on to find the sub-groupings mentioned above.

How does machine learning compare with conventional methods?

Both supervised and unsupervised machine learning have their conventional statistical analogues. For supervised learning it is regression - where we fit a curve to a set of data points, or else classify data points on either side of a function, such as with our weighted tally. One machine learning version of this is the Random Forests algorithm which iteratively finds a ‘tree of features’ - such as a ‘decision tree’ - that best fit your data to the classifications you provide, and allows you to use this decision tree to classify any new example.

For unsupervised learning the analogue is cluster analysis - finding clusters in unlabelled data using techniques such as Principal Components Analysis, which is a computationally intensive algorithm that aims to find the main groups of features that best describe your data, but may make assumptions about the distribution of features.

A simple machine learning algorithm to do cluster analysis is the K-means algorithm, which is an iterative way of finding centres of clusters. Another recent, and extremely powerful machine learning technique is T-SNE, which makes fewer assumptions about the data than Principle Components Analysis or K-means - but is more complicated than K-means.

Why do We Need Machine Learning?

If we have conventional statistical techniques, then why do we need machine learning? Firstly, because conventional techniques do not scale well as you increase the number of examples and features in each example. For instance, an image from a camera might be 1000 x 1000 pixels = 1 million pixels, each of which counts as a "feature".

This is far too many features on which to use statistical techniques, but machine learning techniques routinely handle such images. Secondly, while one could argue that getting a computer to play chess, or the Chinese board game, ‘Go’, is just a matter of scaling up a statistical technique, another view is that these tasks are so far beyond anything that statistics could do that it is a different kind of functionality altogether that machine learning provides.

A good task for machine learning would have:

1) Many features

2) Multiple examples

3) No explicit way of modelling, classifying or grouping them

(but yet, we believe there is one…).

What are Artificial Neurons?

Neural networks are a class of machine learning technique inspired by neurons in the brain. A brain neuron is like a little machine with many electrical inputs – synapses - and one electrical output - the axon. If the sum of electrical impulses from the input synapses exceeds a particular threshold then the neuron "fires" an electrical pulse down its axon - otherwise it stays silent.

The Human Brain Contains c100 Billion Neurons The electrical inputs can come from the axons of other neurons, or else from sensory nerves, such as touch sensor cells in the skin or light sensor cells in the retina. The axon is split at the end and can be connected to multiple synapses in other neurons, or it may connect to motor nerve cells that cause muscles to contract, making us stand up or speak - or to glands that make us sweat or fret.

To get an idea of numbers: there are in the order of a hundred billion neurons in a human brain. Each neuron receives synaptic inputs from around ten thousand other neurons or sensor cells, and the axon of each neuron splits and connects to some thousands of other neurons or motor nerve cells. A human brain can contain a few hundred trillion synapses.

McCulloch & Pitts' Neuron Model

Back in the 1950's McCulloch and Pitts made a simple model of a neuron as a box having binary-valued inputs (i.e. each input could be set to 0 or to 1), a threshold number, which they could set, and a binary-valued output.

They also added weights, which they could set and multiply the inputs by. If the sum of the weighted inputs was greater than the threshold then this artificial neuron would output a binary 1 ("fire"), otherwise it would output a 0 ("stay silent").

Several extensions were made to the model but "threshold sum of weighted inputs" is still a convenient way to think of it.

How this Works for Artificial Neurons

By changing the weights and threshold we can change the behaviour of the artificial neuron in response to its inputs. For example, the first input might indicate whether delivery address was changed, and we could set a large weight for it.

This means that if the input is 1 ("yes, the delivery address was changed") then its weighted value will contribute a lot to the sum (score) and the neuron will more likely fire.

Another input might indicate whether this is a returning customer. We might give it a negative weight, meaning that if the input was set to 1 ("yes, this is returning customer") then its weighted value would reduce the sum and tend to inhibit the neuron from firing.

So, What are Artificial Neural Networks?

An artificial neural network is a set of artificial neurons connected to each other, and to external inputs and outputs. Often, they are wired up in layers: external inputs (e.g. pixels from a camera, or features from our online transaction) feed into every one of the neurons in the bottom layer, each of which fires (or not), depending on whether its sum of weighted inputs is greater than its threshold.

The outputs of the bottom layer neurons are then used as inputs to the neurons in the next layer, each of which fires or not according to its sum of weighted inputs and threshold. These outputs then feed into the neurons in the next layer, and so on, until we reach the top, output layer of neurons, which give the response of the whole network to the inputs we showed it.

Each weight and threshold of each neuron in the network can be set independently of the others. "Training" such a network means changing all the parameters - weights and thresholds - until the network provides the desired outputs in response to the given inputs.

How Complex are Artificial Neural Networks?

To get an idea of numbers, the AlphaGo machine that is the current world champion of that Chinese board game, ‘Go’, has about twelve layers with twenty thousand artificial neurons in each layer, each neuron being fully connected to the neurons in the layer below.

A network capable of recognizing handwritten digits might have three, fully connected layers of ten neurons each, the ten neurons of the bottom layer would each be connected to all the pixels in the 20 x 20 pixel images it is shown.

How do you set the weights and thresholds?

In supervised learning, your examples are labelled and you know the output you should expect for a given input. If the network gets it wrong, you can use the difference between expected and actual output to change the parameters of the top layer neurons so that if the network sees the same example again it will give a more correct response.

Then you go down to the next layer and change parameters in its neurons so they will give a response that is more likely to make the layer above fire correctly. You continue changing the parameters of the layers of neurons back from the top in this way until you reach the bottom layer. Then you repeat with another training example, or else run several training examples in batches and use their average error.

You only change the parameters by a small amount each time so that you add to the experience of previous training examples rather than overwriting them completely. This technique of using the output error to drive changes in neural layers, working back from the output layer is called "error back propagation." You might need to run this algorithm several hundreds or thousands of times over the whole training set before the neural network learns the task.

If you make a network sample just one part of the image and repeat it in overlapping windows all over the image you get a "convolution network" that can classify objects even if they appear in different parts of the image. The lower layers of the AlphaGo machine are convolution networks that can recognize sub-patterns of pieces wherever they appear on the Go board.

The "Self-Organizing Map" Model

For unsupervised learning a simple technique is to (conceptually) place neurons in a grid and randomly set their weights - no thresholds. Then you present a training example to each neuron in the grid and pick the one with the largest weighted sum and "reward" it and its immediate neighbours in the grid so they would respond even more strongly to that example, and you "punish" the neurons further away so they would respond less strongly to that example.

If you repeat this many times for all examples you find that some neurons in the grid tend to become the centres of clusters of your training examples. And you can examine the weights of these neurons to see what the features of these clusters are. This machine learning model is a called Kohonen Network and is an example of a "Self-Organizing Map", it is a neural network method comparable to other machine learning methods such as K-means and T-SNE, mentioned above.

Sounds great, what are the gotchas?

While machine learning scales better than conventional statistical techniques, it is still computationally very expensive and there are limits to the number of parameters you can use.

There are ways of getting around these - by doing training in parallel on several computers, or performing computations on remote computers in a "cloud", or by using special hardware.

One simplified version of Moore's Law says that the "capability" of computers doubles every eighteen months This means, for example, the function of: clock speed, number of transistors per microchip, amount of memory and number of operations per second. Moore made this prediction in 1965 and it has held true ever since – so, we can expect to be able to use more parameters for our machine learning algorithms in the future.

Underfitting and Overfitting

But there are some theoretical problems: there is "underfitting", and then there is "overfitting", which is not to be confused with "local minima". These three can apply to all machine learning algorithms.

Underfitting, also known as "bias", is when the task is too complex for the machine learning model you are trying to use. For example, you could not use a 3-layer network to play ‘Go’.

One solution is to add extra parameters. In a neural network you could add more input features, each with their own weights, or you could add extra layers of neurons.

If you add too many parameters then you run into the danger of overfitting, also known as "variance", or "generalization error", whereby the machine learning algorithm is not generalizing enough from its training set.

It is like a child with an excellent memory who has learnt some sums at school by rote. If you ask the child one of the sums it knows it will give you the correct answer but if you ask the child a sum it doesn't know you might get a crazy answer.

It would be better if the child did not rely on its memory and instead generalized and learnt the rules of arithmetic from the example sums it was given.

The "cost function" of a machine learning system is the average of the errors when you run it against a test set of examples.

Hopefully, every time your training algorithm tweaks the parameters the cost goes down. But you can get into a local minimum where the cost is quite high but any small tweak you make only increases it.

Suppose you are on a mountain in the mist and your (dangerous) plan is to get down by following a stream, only to find it ends up in a little lake or "tarn" at a flat place on the mountainside. If you are wet up to your knees in the middle of the tarn in the mist then any step you take is uphill - i.e. you are in a ("local") minimum of the cost function, but still halfway up the mountain, instead of wringing out your socks in a warm hut at the bottom.

There are several ways round this - you can re-run your learning algorithm with parameters initialized randomly so you are more likely to find a better minimum. This is like starting from several points on the mountainside, trying to follow several streams to see if one of them will get you down. Or else, you could broadly follow your stream but make random forays into the mist in the hope of finding a better one to follow downhill.

The take-home message is that one should be aware of the problems of underfitting, overfitting and local minima, but there are ways of getting around them.

Where do I go from here?

There are many types of machine learning. If someone talks to you about their favourite type of ML - e.g. the Support Vector Machine (SVM). You might ask whether it is a neural network model? - Answer: No, but SVM is an iteratively trained Machine Learning model.

Is it supervised or unsupervised?

Answer: The SVM algorithm uses training examples with labels, so it is a supervised learning model.

And you could ask how it gets around the problems of underfitting, overfitting and local minima.

Answer: The SVM algorithm can use different "kernals" to improve its behaviour on different data sets and it has as many parameters as data points so underfitting should not be a problem. It can "regularize" its parameters to reduce overfitting, and keep aside some of its training examples in a "cross validation set" to check for it. And if its kernel function satisfies "Mercer's Theorem" then there are no local-minima.

If you encounter a neural network model, then you could compare the type of its neurons with McCulloch and Pitts "thresholded sum of weighted inputs" model, compare its architecture with a layered network, and its learning algorithm with error back-propagation. e.g. WISARD is a classifier that uses small RAM nodes instead of McCulloch and Pitts neurons. A single layer of these oversamples a video image like a convolution network. The parameters are memory locations in the RAMs, and training is the selective setting of these in response to the training images. The output of a trained WISARD classifier is the number of RAM nodes that respond to the presented image...

Well, you might not know all the terms mentioned, but you know enough to ask more questions.

The YouTube channel ‘3Blue1Brown’ has a wonderful introduction to neural networks starting with: https://www.youtube.com/watch?v=aircAruvnKk

After that, if you are mathematically- and computer-minded, then the free Coursera Stanford Course on Machine Learning by Prof Andrew Ng is highly recommended by many reviewers. Setting up the coursework examples was a bit tricky, but after that I found it gave an excellent theoretical grounding in supervised learning and the K-means unsupervised model. However, I felt the course fizzled out somewhat at the end.

The following website reviews this and several other courses:

https://medium.freecodecamp.org/every-single-machine-learning-course-on-...

If you just want to play then http://playground.tensorflow.org is a fun visual,

interactive neural network. The many options make more sense if you have already watched the YouTube series from ‘3Blue1Brown’.

Ethical issues arising from machine learning are a hot topic. This article: https://www.weforum.org/agenda/2016/10/top-10-ethical-issues-in-artificial-intelligence/ by the World Economic Forum gives an accessible summary.

Bio

Amongst a host of other specialities, Adrian Redgers has deep expertise and abiding interest all things pertaining to Articficial Intelligence & machine learning; he is based in London and programmes computers for financial institutions in the UK and mainland Europe. He has degrees in Applied Mathematics and Artificial Intelligence, and spent 5 years researching Artificial Neural Networks at Imperial College, London. His other interests include minefields, soup and public speaking.
More about Adrian