Introduction To Machine Learning

In this lecture. . .

- The rules for classical mathematical modelling
- How machine learning is different
- The principal techniques of machine learning

Introduction

A definition from Wikipedia:

Machine learning is a field of computer science that uses statistical techniques to give computer systems the ability to 'learn' (e.g., progressively improve performance on a specific task) with data, without being explicitly programmed.

Take data and statistical techniques, throw them at a computer and what you have is machine learning.

Learning Is Key

Except that there's more to it than that. And there's a clue in the title of the subject, in the word 'learning.'

In traditional modelling you would sit down with a piece of paper, a pencil and a single malt and. . .

- Example: You would write down some differential equation that captures how you believe that inflation will respond to a change in interest rates. As inflation rises so a central bank responds by increasing interest rates. Increasing interest rates cause. . .

You might 'eyeball' the data a bit, maybe do a small amount of basic statistics, but mainly you'd just be using your brain.

Whatever the problem you would build the model yourself.

In machine learning your role in modelling is much more limited.

You will decide on a framework, whether it be a neural network or a support vector machine, for example, and then the data does the rest.

The algorithm *learns*.

A Little Bit Of History

Noughts and Crosses

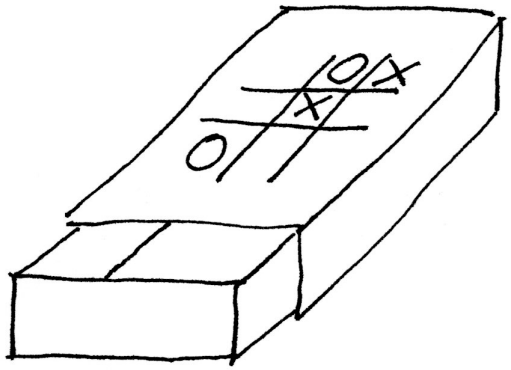
Let me introduce Donald Michie.

- Donald Michie had worked on cyphers and code cracking with his colleague Alan Turing at Bletchley Park during the Second World War

- After the war, as a professor in Edinburgh, he turned his attention to the problem of training a computer to play the game of Noughts and Crosses, a.k.a. Tic Tac Toe.

- Well, not so much a computer as an array of matchboxes. Three hundred and four matchboxes, laid out to represent the stages of a game of Noughts and Crosses.

For example a matchbox might have the following drawn on it:

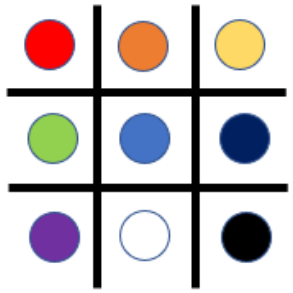


In this it is X's turn to play. There is clearly a winning move here. (Actually two of them.) I know that, and you know that, but if the computer were playing would it also know?

Traditionally one might programme a computer with all the possible states of the game and the best next move in each case. That's doable for N&Cs but most definitely is not with any non-trivial game.

What Professor Michie did instead was to fill each of his 304 matchboxes with coloured beads, each bead representing one of the empty cells in the grid. (He took advantage of symmetries or there would have been even more than 304 matchboxes.)

- Shake the matchbox, remove one of the beads at random. This gives us the next move
- The game continues, and we move from state to state and matchbox to matchbox
- If we win the game then each matchbox used in that game gets rewarded with extra beads of its chosen colour, and if we lose then a bead of that colour gets taken away



Eventually the matchboxes fill up with beads representing the most successful action at each state.

The machine, and Michie called it MENACE, for Machine Educable Noughts And Crosses Engine, has learned to play Noughts and Crosses!

AlphaGo

How Google DeepMind programmed a computer to learn to play Go. . . And it promptly went on to beat Lee Sedol, a 9-dan professional Go player.

AlphaGo used a variety of techniques, among them neural networks. Again it learned to play without explicit programming of optimal play. Curiously not all of that play was against human opponents. . . it also played against itself.

- If you used matchboxes to represent states then you'd need a lot more than 304 of them, you'd probably have to fill the known universe with matchboxes!

I would strongly recommend watching the creatively titled movie 'The AlphaGo Movie' to watch the faces of everyone involved. Lee Sedol started moderately confident, became somewhat gutted, and finally resigned (literally and psychologically). Most of the programmers were whooping for joy as the games progressed. Although there was one pensive-looking one who looked like he was thinking "Oh. . . my. . . God. . . what. . . have. . . we. . . done?"

- At times the computer made some strange moves, moves that no professional Go player would even have considered. During the movie there was discussion about whether this was a programming error, lack of sufficient training, or a stroke of computer genius. It was always the last of these.

Key Methodologies

Principal categories

- **Supervised Learning:**
  - You are given both inputs and outputs
  - The input would be a vector, i.e. more than one variable
  - We can use supervised learning for regression, predicting numerical values for outputs for any new data points we input (e.g. input temperature and time of day, and predict electricity usage)
  - Or we can use it for classification. (We have pictures of dogs and they are classified as Hound, Toy, Terrier, etc. We train our algorithm on these pictures and we would want it to correctly classify any new picture we give it)

- **Unsupervised Learning:**
  - The data is not labelled
  - We only have inputs, not outputs
  - The algorithm finds relationships or patterns for you. (E.g. We show the computer pictures of dogs, now the computer groups or clusters them together according to whatever features it deems most important. Perhaps it will come up with Hound, Toy, Terrier,...or perhaps black, brown, white,...or something completely different that humans don't even notice)

  Unsupervised learning might at first seem strange, but you'll get the picture if I said to you "Here are the contents of my desk drawer. Please sort them out."

- **Reinforcement Learning:**

  – An algorithm learns to *do* something

  – It is rewarded for successful 'behaviour' and punished for unsuccessful behaviour

  The above-mentioned MENACE was a simple example of this, the rewards and punishments being the addition or subtraction of beads to or from the matchboxes.

  Computers learning to play board games or video games from the 1980s seem to get a lot of publicity. That would usually involve reinforcement learning.

Principal techniques

- K **Nearest Neighbours:** K nearest neighbours (KNN) is a **supervised-learning technique** in which we measure distances between any new data point (in any number of dimensions) and the nearest K of our already-classified data and thus conclude to which class our new data point belongs. It can also be used for regression.

- K **Means Clustering:** K means clustering (KMC) is an example of an **unsupervised-learning technique**. We have lots of data in vector form (representing many dimensions of information about each point) and we measure the distance of each data point from K centroids. Then we find the optimal position for these centroids that gives us the best division into the K classes.

- **Naïve Bayes Classifier:** Naïve Bayes classifier (NBC) is a **supervised-learning technique** that uses Bayes Theorem to calculate the probability of new data points being in different classes. The 'naïve' bit refers to the assumptions made, which are rarely true in practice but that doesn't usually seem to matter.

- **Regression Methods:** Regression methods are **supervised-learning techniques** that try to explain a numerical dependent variable in terms of independent variables. You'll probably know of linear regression at least. But you can go much further with more complicated types of regression.

- **Support Vector Machines:** A support vector machine (SVM) is a **supervised-learning technique**, one that divides data into classes according to which side of a hyperplane in feature space each data point lies.

- **Self-organizing Maps:** A self-organizing map (SOM) is an **unsupervised-learning technique** that turns data in many dimensions into nice, typically two-dimensional, pictures for visualizing relationships between data points. Imagine a chess board and into the squares you put data points with similar characteristics. You don't specify those characteristics, they are found as part of the algorithm.

- **Decision Trees:** This is a **supervised-learning technique**. A decision tree is just a flowchart. "How many legs does it have? Two, four, more than four?" Four. Next, "Does it have a tail? Yes or no." Yes. And so on. Like a game of 20 Questions. But can the machine learn and improve on how humans might classify? Is there a best order in which to ask the questions so that the classifying is done quickly and accurately? It can also be used for regression.

- **Neural Networks:** A neural network (NN) (sometimes with 'artificial' in front) is a type of machine learning that is meant to mimic what happens in the brain. An input vector is transformed, typically by multiplying by a matrix. So far, so linear. Then the result is put into some non-linear 'activation function.' That's one layer. But that result then usually goes into another layer, and so on. In this form it's really just a non-linear function of the original data. But it can get more complicated by having the data not just going in one direction but also looping back on itself. The parameters in the neural network are found by training. It can be used for either **supervised** or **unsupervised learning**.

Classical Mathematical Modelling

I want to explain, briefly, how machine learning is different from classical mathematical modelling.

There's a well-recognised, and well-trodden, path that (we) mathematical modellers go along when faced with a new problem. It goes something like this. . .

First decide on your variables

- Variables come in two main types, independent and dependent

- Independent variables represent the domain in which your problem is specified. A common group of independent variables would be spatial and time coordinates, x, y, z, and t

- Contrast this with the dependent variables, these are the things you are solving for. (Consider the oscillation of a violin string. The dependent variable would be distance by which the string is perturbed from rest)

- You really want the minimum of variables to specify the system and its behaviour

- **Rule 1:** Einstein said something along the lines of "Make things as simple as possible, and no simpler."

- **Rule 2:** PW said "If you are going to make things more complicated then do so one new feature at a time"

- That's a big difference to machine learning in which one tends to throw in everything at the start

The goal at this stage in classical mathematical modelling is to get to the first **'minimally interesting non-trivial problem.'**

Figure out what are the main 'drivers' in your system

What drives your system? Are there forces pushing and pulling something? Are molecules bouncing off each other at random? Do buyers and sellers move the price of milk up and down?

Quantitatively accurate models: If you are lucky you'll have some decent principles to base your model on, such as conservation of something. The violin string moves according to Newton's second law, the force caused by the bend in the string makes the string move.

Toy models: But maybe you have to fall back on some common sense or observation. To model the dynamical system of lions and gazelles (the former like to eat the latter) then you would say that the more lions there are the more they breed, ditto for gazelles, and the more gazelles there are the more food for lions, from which it is possible to write down a pretty, but not necessarily quantitatively accurate, model.

Parameters

Your model will probably need you to give it values for some parameters. Again if you are lucky you might be able to get these from experiment; the acceleration due to gravity, or the viscosity of a fluid. And perhaps those parameters will stay constant. But maybe the parameters will be difficult to measure, unstable, or perhaps not even exist, the volatility of a stock price for example.

These are issues that the mathematical modeller needs to address and will make the difference between a model with reliable outputs and one that is merely useful for explanation of phenomena.

Decide on your type of mathematics

There are many branches of mathematics, some are more easily employed than others when it comes to practical problems.

Discrete mathematics, or continuous mathematics? Will you be using calculus, perhaps ordinary or partial differential equations? Or perhaps the model is going to use probabilistic concepts. If you are trying to figure out how to win at Blackjack then clearly you'll be working with discrete values for the cards and probability will play a major role. As will optimization. Change to poker and game theory becomes important.

Machine Learning Is Different

Machine learning could not be more different if it tried. Almost nothing in the above carries over to machine learning.

- One big difference is the role of data. In classical mathematical modelling you might not have any data.

Say you have a problem in fluid mechanics then you build your model on conservation of mass and momentum. You'd come up with a model while sitting in your office with nothing more than a pencil and some paper. There would be but a few parameters (fluid density and viscosity) and you are done.

If this were a machine-learning problem then you would show the machine a great deal of data, perhaps in the form of movies of flow past various shapes, and let it do its thing.

All being well the machine would learn and somewhere deep inside it would have indirectly hit on something like the Navier–Stokes equation. You certainly wouldn't have specified any drivers, conservation laws, etc.

Type of mathematics?

The role of type of mathematics would become the choice of machine-learning scheme. Is the problem best approached via supervised or unsupervised learning? Would it be a self-organizing map or K means clustering that gave the most reliable results?

Summary

Please take away the following important ideas

- There are three main categories of machine learning: Supervised; Unsupervised; Reinforcement
- Within these categories there are many techniques
- Machine learning is very different from classical mathematical modelling

