Course Introduction

Most machine learning methods are ultimately ways to set some numbers in a function that we don't want to set by hand. For some students, fitting functions to data may not sound like the glamorous endeavour of artificial intelligence that you signed up for. However, the fundamentals we learn in this course are both used in many practical deployed applications, and the basis of some more blue-sky research that's reaching for the grander goals of AI.

As an example application[1], many cameras can identify faces in an image, and will auto-focus on them. We could try to write a function by hand that takes the pixel-values in an image and identifies faces. Perhaps two nearby dark dots, each with white on either side, strongly suggest eyes. We could write a rule to find this feature and report a face when we see it. Similarly we might be able to write nose, lip, or ear detectors. However, making a system that works robustly across ethnicities, alternative poses, with different hair-styles and accessories, and under all possible lighting conditions, turns out to be hard. The first widely-available cameras with face detectors used a "Viola–Jones object detector", which you could think of as a piece of code with a very large number of "magic values": numbers that can be tweaked to alter which objects the detector responds to. These numbers have been fitted by a machine learning algorithm to perform well on a large dataset of faces. The camera in your phone might not use a Viola–Jones detector, but it will definitely use a face detector that was trained with a machine learning algorithm.

As another example, most of our email is now classified by computers as spam, phishing, virus, or ok. It would be possible to write a function by hand to attempt this task. There are phrases that will never occur in an email I want to read, that I could black-list (e.g., *"IT Help Desk"*). I could also come up with some score-based rule, where I discard the email if too many suspicious words occur. However, we can use machine learning to bypass a lot of the manual work, *and* we get better classifiers.

Machine learning systems are big business. The most lucrative task is the automatic choice of which adverts to display. However, machine learning is also used by internet businesses to make recommendations to more willing customers, to manage stock, and to detect anomalies that need attention. Machine learning is also an important component of vision and natural language processing systems, which are used to intelligently index and process the posts and photos stored on web services.

Personally, I'm excited about broader uses of machine learning, such as scientific discovery. Analysis pipelines for gravitational waves, dark matter, and particle physics all use methods developed in machine learning. The potential impact of machine learning on society is also far more wide-reaching than the computer applications we use. You have probably heard of active research on self-driving cars or personalized medicine.

1 Principles of the course

The aim of this course is to cover a coherent(-ish) set of machine learning fundamentals. We certainly won't cover every machine learning algorithm, or even every theoretical approach. However, you should leave the course with the ability to derive and implement both the methods we cover, and novel variants. After taking this course you should also know how to approach selecting, critically comparing, and evaluating different machine learning methods. Finally you should appreciate both what these machine learning methods can achieve, and their pitfalls.

We will spend a lot of time on probabilistic models and Bayesian methods, which reflects the research approach of many in the School. This theme continues in the Probabilistic Modelling and Reasoning (PMR) course next Semester.

1. Acknowledgement: the opening example is heavily influenced by Geoff Hinton and Sam Roweis's teaching.

The main aim of the first week is to get everyone in the class to the stage where you understand a particular model: "linear regression with basis functions". From this foundation we will study how to evaluate and select amongst different models, and alternative principles for learning these models. We will extend linear regression and the learning principles to deal with different types of data, and into more flexible neural-network based systems. So it's important that you follow the linear regression material!

2 Learning functions for the example applications

Classifying text, for example to mark spam emails, is an example of an application where learning a simple *linear* function gets us a long way. We can extract a *feature vector* $\mathbf{x}$, a vector of numbers indicating the presence or count of particular words in an email. Then construct a linear function:

$$f(\mathbf{x}) = w_1 x_1 + w_2 x_2 + \ldots + w_D x_D = \sum_{d=1}^{D} w_d x_d = \mathbf{w}^\top \mathbf{x}. \tag{1}$$

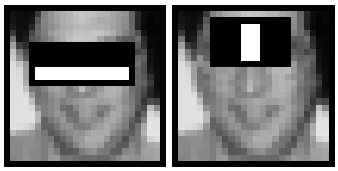
If we set each weight w_d to be positive for 'spammy' words, and negative for words we like to see, then it might be reasonable to mark an email as spam if $f(\mathbf{x}) > 0$ for that email's feature vector.

There are quite a few details that we would need to get right for a working system however. The business cost of discarding legitimate email is larger than that of keeping spam email. In general we need to understand the theory for turning the predictions of a system into an action or decision. We also need to decide which words to look for in each email. Or, if we note the presence/absence of all possible words seen on the internet, we need to deal with the computational cost and statistical problems of fitting a model with $D > 10^6$ parameters. Finally, the sorts of emails that we and spammers send change all the time, and the spammers changes are likely to be adversarial (designed to break our system).

In other applications, linear classifiers will get us nowhere. The gray-scale value of a particular pixel, for example the 582nd pixel in an image, tells us nothing (for practical purposes) about the content of the image. Thus we can't sensibly give a single pixel x_d a positive or a negative weight w_d in an image classifier. A single linear function of the pixels $\mathbf{w}^\top \mathbf{x}$ isn't going to solve interesting image recognition tasks.

One way to deal with complicated data like images is to exploit human expertise and intelligence. "Feature engineering" is the art of converting data into numerical features that make machine learning methods work well. In practice, feature engineering is an important part of making many machine learning applications work. It's also a goal of machine learning to get rid of the need for feature engineering!

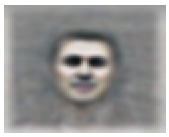
The Viola–Jones classifier extracts only simple and cheap-to-compute features from an image. The following images come from the original paper (Viola and Jones, 2001):



If we add up the pixels in the white rectangle, and subtract the sum of the pixels in the black rectangle, we get a large value when the black rectangle is in a relatively dark region. Thus the first feature takes on large values when over the dark shadows in the eye region. The second feature takes on large values when seeing a bright stripe from the nose. The Viola–Jones method decides both where to place these rectangles, and how to combine the resulting feature values. The first system made these choices by learning from 4,916 example face images, aligned by hand.

Learning from labeled examples, as in the examples above, is often called *supervised learning*. *Unsupervised learning* means learning how to represent data using unlabeled examples. For example, Le et al. (2012), built a large-scale version of a function called an "auto-encoder", with a billion free parameters. The free parameters were fitted to encode frames of YouTube videos. We'll talk about auto-encoders later in the course. For now, they are a function that take an image, compute a series of intermediate vectors, and output another image. The auto-encoder's parameters are fitted so that given an input image, the function outputs an image as close as possible to the input image.

One element of one of the intermediate vectors in Le et al.'s system took on large values whenever the input image contained a face. The system wasn't told which training images contained faces, it just turned out that representing faces was a side-effect of the learning algorithm. One could ask whether the network has really captured what a face is, or is luckily "classifying" faces just on this particular data. Le et al. optimized an image (starting with random pixel values) to make the intermediate face-detecting element as large as possible. The resulting image was as follows:



The element learned "unsupervised" by the system really did respond to faces! Another element of one of the vectors 'liked' cats. (See the paper for cat images.)

A team at OpenAI made a similar discovery with text in 2017. They found a "sentiment neuron", an intermediate number computed in a neural network, that corresponded to the sentiment of some text. The neural network model (an mLSTM, developed here in Edinburgh), was told nothing about sentiment, it was just given a large collection of unlabelled text.

3 A rant about least squares, and the nature of machine learning

If you want to produce some pretty images, or do some computer vision, there are a lot of neural network packages and demonstrations out there. With a bit of Python and a powerful computer you can do a lot of amazing things very quickly. I'll end this lecture note with some motivation for learning the theoretical background.

The text below is an abridged excerpt from: *Numerical Methods that (usually) Work*, by Forman S. Acton, Spectrum Paperback 1990 (original edition 1970), p253. I've put an unabridged excerpt online.

> ''Whenever a person eagerly inquires if my computer can solve a set of 300 equations in 300 unknowns... The odds are all too high that our inquiring friend... has collected a set of experimental data and is now attempting to fit a 300-parameter model to it — by Least Squares! The sooner this guy can be eased out of your office, the sooner you will be able to get back to useful work — but these chaps are persistent... you end up by getting angry and throwing the guy out of your office.
>
> There is usually a reasonable procedure. Unfortunately, it is undramatic, laborious, and requires thought — which most of these charlatans avoid like the plague. They should merely fit a five-parameter model, then a six-parameter one... Somewhere along the line — and it will be much closer to 15 parameters than to 300 — the significant improvement will cease and the fitting operation is over. There is no system of 300 equations, no 300 parameters, and no glamor.
>
> The computer center's director must prevent the looting of valuable computer time by these would-be fitters of many parameters. The task is not a pleasant one, but the legitimate computer users have rights, too... the impasse finally has to be broken by violence — which therefore might as well be used in the very beginning."

I like the book that the excerpt above comes from.[2] Given how much I was enjoying the book, I paused for thought when I read the excerpt above. I've been 'guilty' of fitting 300 parameters by variants of least squares, and machine learning researchers sometimes now fit neural networks with *billions* of parameters. Are machine learning researchers 'charlatans' that should be doing something simpler and less glamorous?

It is tempting to dismiss the rant as an out-dated statement from a time when computers were slow and expensive. We have far more powerful computers than we had in 1970, including in most of our pockets. Solving a single system of 300 equations doesn't have any significant computational cost worth mentioning. However, as compute power has increased, so has our ambition for glamour. Some machine learning papers now report models that require weeks to fit on clusters of GPUs. Researchers' compute time is still a significant expense… and use of energy. Whether a large machine learning application is a responsible use of resources can still be a legitimate question.

Even ignoring computational cost, Acton also had valid statistical concerns. We can now process much more data than in 1970, but issues with fitting many parameters still exist. After taking this course you should be able to explain the problems that can result from attempting to fit 'too many' parameters, and how to detect and mitigate these problems.

The particular advice, to start with a simple model, and gradually increase its size, is both right and wrong.

A weak student project only tries a large complicated system. A basic requirement of a project is to compare to appropriate baselines, and to test what parts of a system are necessary for it to function. If an application only requires a simple method, fitted with a small amount of data, why do more? So starting with the smallest system that might work and building it up is often a good strategy.

However, the peculiar history of machine learning is that the 'charlatan' approach (using far more parameters than seem statistically justified) has worked well. It can be easier to find parameter settings that work well for large machine learning systems than small ones, because there are often more settings of the parameters that will work well. However, some form of "regularization" (covered in later lectures) is required to avoid the numerical and statistical problems that result from having 'too many' parameters.

4 Is this note examinable?

In general, anything covered in the course might be tested on the exam, except where marked "non-examinable" or "for keen students". However, this general introduction was meant for motivation, and where material is examinable, we will revisit it in future notes. You do *not* have to know the details of the particular Viola and Jones or Le et al. papers mentioned in this note. Although it is always possible an exam could describe details of such a system, and then ask you to apply your knowledge from the course to discuss it.

5 Check your understanding

The main task for the first week is to check whether you should be taking IAML or this class. See the maths and programming background notes and self test. Then make sure you keep up with the linear regression material that comes next.

Please go through the website and look at everything that's available. The forum/annotation system lets you highlight any of it, and ask about anything that's unclear. But *please* read the instructions for the forum system, and post to the correct group.

2. If you have seen MLPR's opening material before, you could entertain yourself by finding a copy of the Acton book and trying its opening exercise about a train track. It's a simple-looking problem that requires a surprising amount of thought and mathematical insight to compute an accurate answer.

Here's a question to get you to revisit a part of this note: A spammer writes a spammy email and downloads a popular spam classifier to see if they'll get away with it. The classifier works as described in this note, and assigns a large and positive score $\mathbf{w}^\top \mathbf{x}$, indicating spam. Come up with two ways that the spammer might fool the classifier. Would there be ways to mitigate these attacks? Would the spammer be likely to be able to further attack your modified system?

6 Further Reading

If you wish to read more now, Chapter 1 of Murphy gives a good overview of machine learning. It's much more detailed and in depth than my first lecture, and almost all relevant for this course.

For keen students: In addition to the citations given in line, here are some references on internet-scale text classification, such as for spam filtering.

- Feature Hashing for Large Scale Multitask Learning, Weinberger et al. (2009).
- Bag of Tricks for Efficient Text Classification, Joulin et al. (2016).