Prologue: A machine learning sampler

YOU MAY NOT be aware of it, but chances are that you are already a regular user of machine learning technology. Most current e-mail clients incorporate algorithms to identify and filter out spam e-mail, also known as junk e-mail or unsolicited bulk e-mail. Early spam filters relied on hand-coded pattern matching techniques such as regular expressions, but it soon became apparent that this is hard to maintain and offers insufficient flexibility – after all, one person's spam is another person's ham![1] Additional adaptivity and flexibility is achieved by employing machine learning techniques.

SpamAssassin is a widely used open-source spam filter. It calculates a score for an incoming e-mail, based on a number of built-in rules or 'tests' in SpamAssassin's terminology, and adds a 'junk' flag and a summary report to the e-mail's headers if the score is 5 or more. Here is an example report for an e-mail I received:

```
-0.1 RCVD_IN_MXRATE_WL       RBL: MXRate recommends allowing
                             [123.45.6.789 listed in sub.mxrate.net]
 0.6 HTML_IMAGE_RATIO_02     BODY: HTML has a low ratio of text to image area
 1.2 TVD_FW_GRAPHIC_NAME_MID BODY: TVD_FW_GRAPHIC_NAME_MID
 0.0 HTML_MESSAGE            BODY: HTML included in message
 0.6 HTML_FONx_FACE_BAD      BODY: HTML font face is not a word
 1.4 SARE_GIF_ATTACH         FULL: Email has a inline gif
 0.1 BOUNCE_MESSAGE          MTA bounce message
 0.1 ANY_BOUNCE_MESSAGE      Message is some kind of bounce message
 1.4 AWL                     AWL: From: address is in the auto white-list
```

[1] Spam, a contraction of 'spiced ham', is the name of a meat product that achieved notoriety by being ridiculed in a 1970 episode of *Monty Python's Flying Circus*.

From left to right you see the score attached to a particular test, the test identifier, and a short description including a reference to the relevant part of the e-mail. As you see, scores for individual tests can be negative (indicating evidence suggesting the e-mail is ham rather than spam) as well as positive. The overall score of 5.3 suggests the e-mail might be spam. As it happens, this particular e-mail was a notification from an intermediate server that another message – which had a whopping score of 14.6 – was rejected as spam. This ‘bounce’ message included the original message and therefore inherited some of its characteristics, such as a low text-to-image ratio, which pushed the score over the threshold of 5.

Here is another example, this time of an important e-mail I had been expecting for some time, only for it to be found languishing in my spam folder:

```
 2.5 URI_NOVOWEL            URI: URI hostname has long non-vowel sequence
 3.1 FROM_DOMAIN_NOVOWEL    From: domain has series of non-vowel letters
```

The e-mail in question concerned a paper that one of the members of my group and I had submitted to the European Conference on Machine Learning (ECML) and the European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD), which have been jointly organised since 2001. The 2008 instalment of these conferences used the internet domain www.ecmlpkdd2008.org – a perfectly respectable one, as machine learning researchers know, but also one with eleven ‘non-vowels’ in succession – enough to raise SpamAssassin’s suspicion! The example demonstrates that the importance of a SpamAssassin test can be different for different users. Machine learning is an excellent way of creating software that adapts to the user.

How does SpamAssassin determine the scores or ‘weights’ for each of the dozens of tests it applies? This is where machine learning comes in. Suppose we have a large ‘training set’ of e-mails which have been hand-labelled spam or ham, and we know the results of all the tests for each of these e-mails. The goal is now to come up with a weight for every test, such that all spam e-mails receive a score above 5, and all ham e-mails get less than 5. As we will discuss later in the book, there are a number of machine learning techniques that solve exactly this problem. For the moment, a simple example will illustrate the main idea.

Example 1 (Linear classification). Suppose we have only two tests and four training e-mails, one of which is spam (see Table 1). Both tests succeed for the

E-mail	x_1	x_2	Spam?	$4x_1+4x_2$
1	1	1	1	8
2	0	0	0	0
3	1	0	0	4
4	0	1	0	4

Table 1. A small training set for SpamAssassin. The columns marked x_1 and x_2 indicate the results of two tests on four different e-mails. The fourth column indicates which of the e-mails are spam. The right-most column demonstrates that by thresholding the function $4x_1+4x_2$ at 5, we can separate spam from ham.

spam e-mail; for one ham e-mail neither test succeeds, for another the first test succeeds and the second doesn't, and for the third ham e-mail the first test fails and the second succeeds. It is easy to see that assigning both tests a weight of 4 correctly 'classifies' these four e-mails into spam and ham. In the mathematical notation introduced in Background 1 we could describe this classifier as $4x_1+4x_2>5$ or $(4,4)\cdot(x_1,x_2)>5$. In fact, any weight between 2.5 and 5 will ensure that the threshold of 5 is only exceeded when both tests succeed. We could even consider assigning different weights to the tests – as long as each weight is less than 5 and their sum exceeds 5 – although it is hard to see how this could be justified by the training data.

But what does this have to do with learning, I hear you ask? It is just a mathematical problem, after all. That may be true, but it does not appear unreasonable to say that SpamAssassin learns to recognise spam e-mail from examples and counter-examples. Moreover, the more training data is made available, the better SpamAssassin will become at this task. The notion of performance improving with experience is central to most, if not all, forms of machine learning. We will use the following general definition: *Machine learning is the systematic study of algorithms and systems that improve their knowledge or performance with experience*. In the case of SpamAssassin, the 'experience' it learns from is some correctly labelled training data, and 'performance' refers to its ability to recognise spam e-mail. A schematic view of how machine learning feeds into the spam e-mail classification task is given in Figure 2. In other machine learning problems experience may take a different form, such as corrections of mistakes, rewards when a certain goal is reached, among many others. Also note that, just as is the case with human learning, machine learning is not always directed at improving performance on a certain task, but may more generally result in improved knowledge.