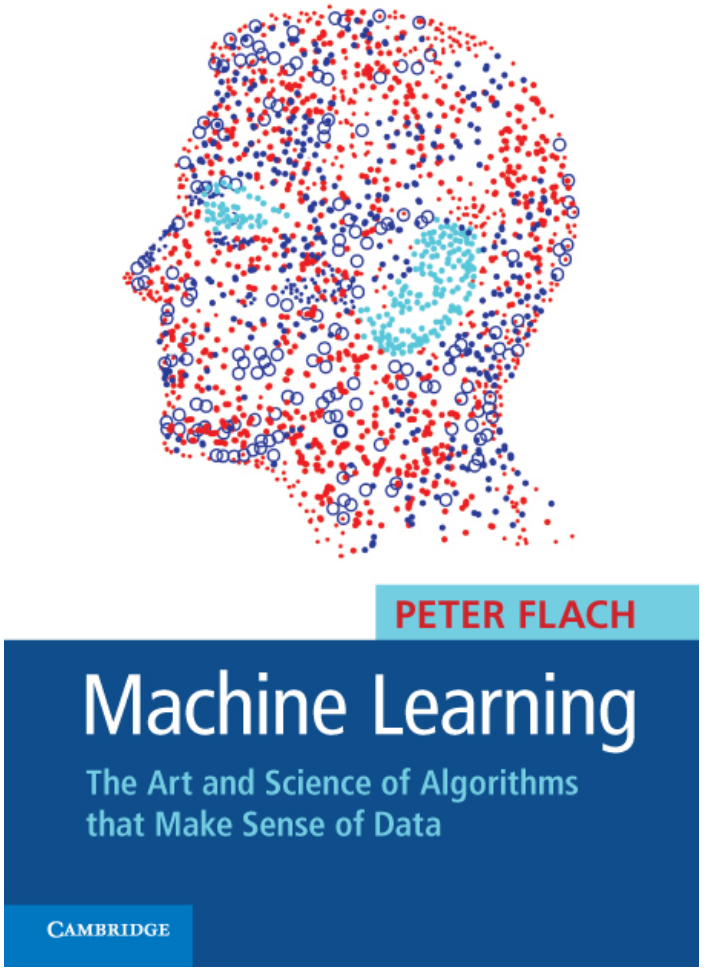
COMS30301: Introduction to Machine Learning

"using the right features to build the right models that achieve the right tasks"

Peter Flach

Department of Computer Science, University of Bristol

Autumn term, 2015


PETER FLACH
Machine Learning
The Art and Science of Algorithms
that Make Sense of Data
CAMBRIDGE

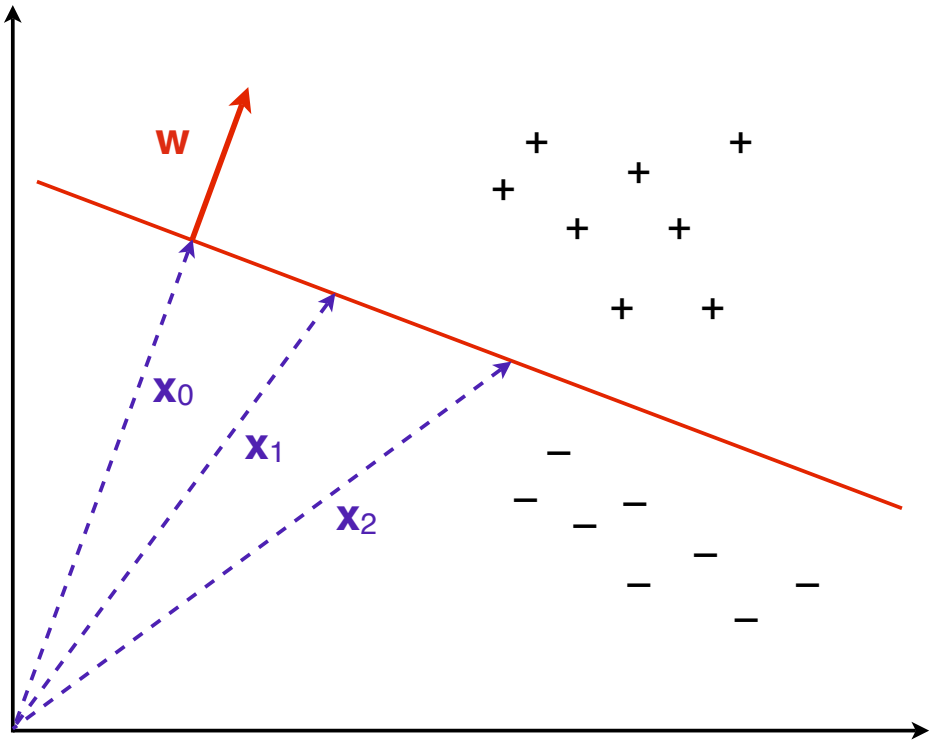
Spam filtering as a classification task

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

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.

Figure 1, p.5

Linear classification in two dimensions



The straight line separates the positives from the negatives. It is defined by $\mathbf{w} \cdot \mathbf{x}_i = t$, where $\mathbf{w}$ is a vector perpendicular to the decision boundary and pointing in the direction of the positives, t is the decision threshold, and $\mathbf{x}_i$ points to a point on the decision boundary. In particular, $\mathbf{x}_0$ points in the same direction as $\mathbf{w}$, from which it follows that $\mathbf{w} \cdot \mathbf{x}_0 = ||\mathbf{w}|| \, ||\mathbf{x}_0|| = t$ ($||\mathbf{x}||$ denotes the length of the vector $\mathbf{x}$).

Homogeneous coordinates

It is sometimes convenient to simplify notation further by introducing an extra constant 'variable' $x_0 = 1$, the weight of which is fixed to $w_0 = -t$.

The extended data point is then $\mathbf{x}^\circ = (1, x_1, \ldots, x_n)$ and the extended weight vector is $\mathbf{w}^\circ = (-t, w_1, \ldots, w_n)$, leading to the decision rule $\mathbf{w}^\circ \cdot \mathbf{x}^\circ > 0$ and the decision boundary $\mathbf{w}^\circ \cdot \mathbf{x}^\circ = 0$.

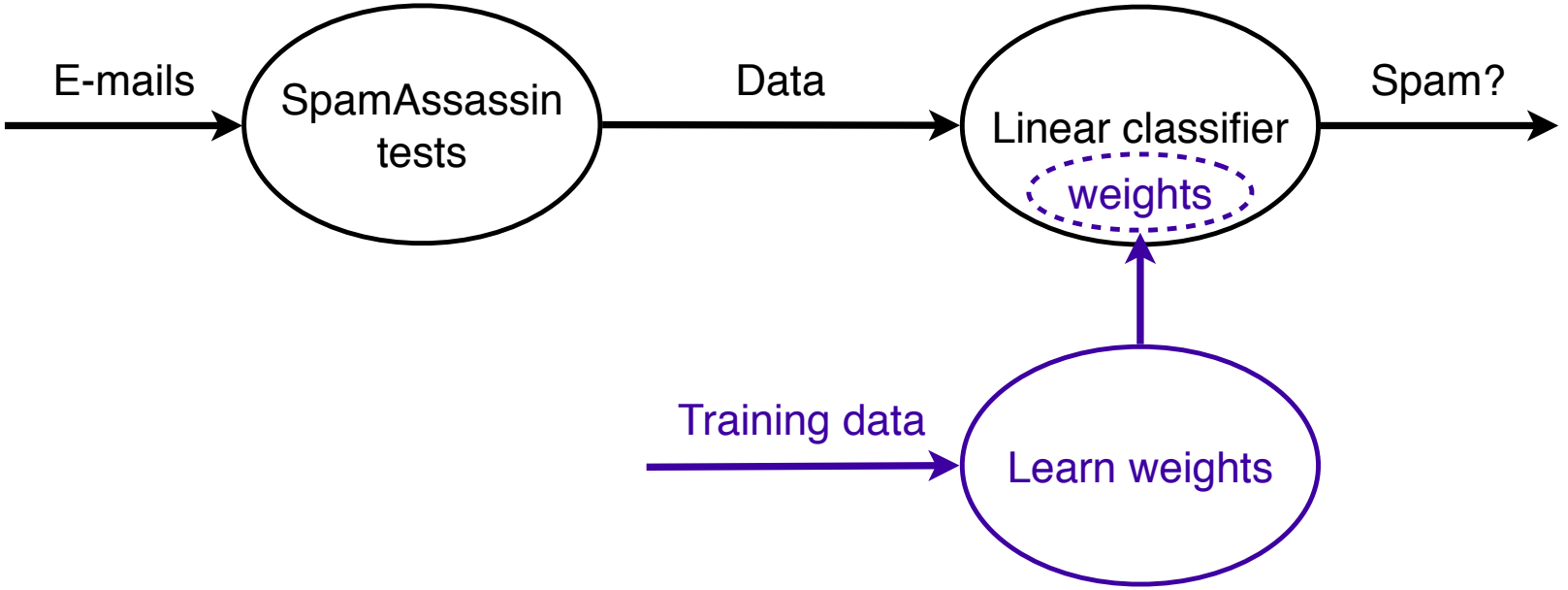
Thanks to these so-called homogeneous coordinates the decision boundary passes through the origin of the extended coordinate system, at the expense of needing an additional dimension.

☞ note that this doesn't really affect the data, as all data points and the 'real' decision boundary live in the plane $x_0 = 1$.

Important point to remember

Machine learning is the systematic study of algorithms and systems that improve their knowledge or performance with experience.

Machine learning for spam filtering



At the top we see how SpamAssassin approaches the spam e-mail classification task: the text of each e-mail is converted into a data point by means of SpamAssassin's built-in tests, and a linear classifier is applied to obtain a 'spam or ham' decision. At the bottom (in blue) we see the bit that is done by machine learning.

Overfitting

Imagine you are preparing for your *Machine Learning 101* exam. Helpfully, Professor Flach has made previous exam papers and their worked answers available online. You begin by trying to answer the questions from previous papers and comparing your answers with the model answers provided.

Unfortunately, you get carried away and spend all your time on memorising the model answers to all past questions. Now, if the upcoming exam completely consists of past questions, you are certain to do very well. But if the new exam asks different questions about the same material, you would be ill-prepared and get a much lower mark than with a more traditional preparation.

In this case, one could say that you were *overfitting* the past exam papers and that the knowledge gained didn't *generalise* to future exam questions.

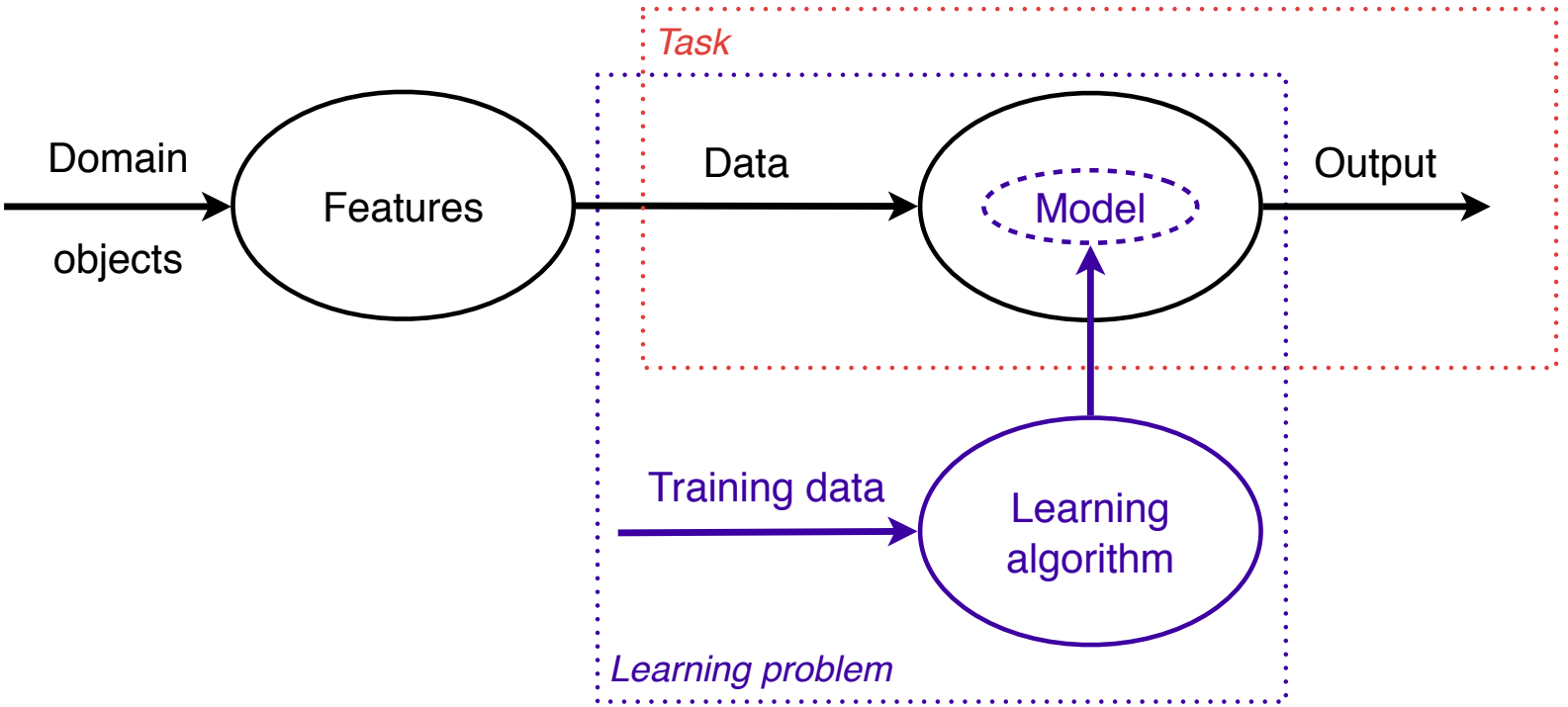
A rule-based classifier

- ☞ if the e-mail contains the word ‘Viagra’ then estimate the odds of spam as 4:1;
- ☞ otherwise, if it contains the phrase ‘blue pill’ then estimate the odds of spam as 3:1;
- ☞ otherwise, estimate the odds of spam as 1:6.

The first rule covers all e-mails containing the word ‘Viagra’, regardless of whether they contain the phrase ‘blue pill’, so no overcounting occurs. The second rule *only* covers e-mails containing the phrase ‘blue pill’ but not the word ‘Viagra’, by virtue of the ‘otherwise’ clause. The third rule covers all remaining e-mails: those which neither contain neither ‘Viagra’ nor ‘blue pill’.

Figure 3, p.11

How machine learning helps to solve a task



An overview of how machine learning is used to address a given task. A task (red box) requires an appropriate mapping – a model – from data described by features to outputs. Obtaining such a mapping from training data is what constitutes a learning problem (blue box).

Important point to remember

Tasks are addressed by models, whereas learning problems are solved by learning algorithms that produce models.

What's next?

1. The ingredients of machine learning
   - Tasks: the problems that can be solved with machine learning
     - Looking for structure
   - Models: the output of machine learning
     - Geometric models
     - Probabilistic models
     - Logical models
   - Features: the workhorses of machine learning
     - Two uses of features
     - Feature construction and transformation

What's next?

Tasks for machine learning

The most common machine learning tasks are *predictive*, in the sense that they concern predicting a target variable from features. .

- ☞ Binary and multi-class classification: categorical target
- ☞ Regression: numerical target
- ☞ Clustering: hidden target

Descriptive tasks are concerned with exploiting underlying structure in the data.

Looking for structure I

Consider the following matrix:

$$\begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 2 & 2 & 2 \\ 0 & 0 & 0 & 1 \\ 1 & 2 & 3 & 2 \\ 1 & 0 & 1 & 1 \\ 0 & 2 & 2 & 3 \end{pmatrix}$$

Imagine these represent ratings by six different people (in rows), on a scale of 0 to 3, of four different films – say *The Shawshank Redemption*, *The Usual Suspects*, *The Godfather*, *The Big Lebowski*, (in columns, from left to right). *The Godfather* seems to be the most popular of the four with an average rating of 1.5, and *The Shawshank Redemption* is the least appreciated with an average rating of 0.5. Can you see any structure in this matrix?

Looking for structure II

$$\begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 2 & 2 & 2 \\ 0 & 0 & 0 & 1 \\ 1 & 2 & 3 & 2 \\ 1 & 0 & 1 & 1 \\ 0 & 2 & 2 & 3 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix} \times \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- The right-most matrix associates films (in columns) with genres (in rows): *The Shawshank Redemption* and *The Usual Suspects* belong to two different genres, say drama and crime, *The Godfather* belongs to both, and *The Big Lebowski* is a crime film and also introduces a new genre (say comedy).
- The tall, 6-by-3 matrix then expresses people's preferences in terms of genres.

Looking for structure III

☞ Finally, the middle matrix states that the crime genre is twice as important as the other two genres in terms of determining people's preferences.

Table 1.1, p.18

Machine learning settings

	Predictive model	*Descriptive model*
Supervised learning	classification, regression	subgroup discovery
Unsupervised learning	predictive clustering	descriptive clustering, association rule discovery

The rows refer to whether the training data is labelled with a target variable, while the columns indicate whether the models learned are used to predict a target variable or rather describe the given data.

What's next?

Machine learning models

Machine learning models can be distinguished according to their main intuition:

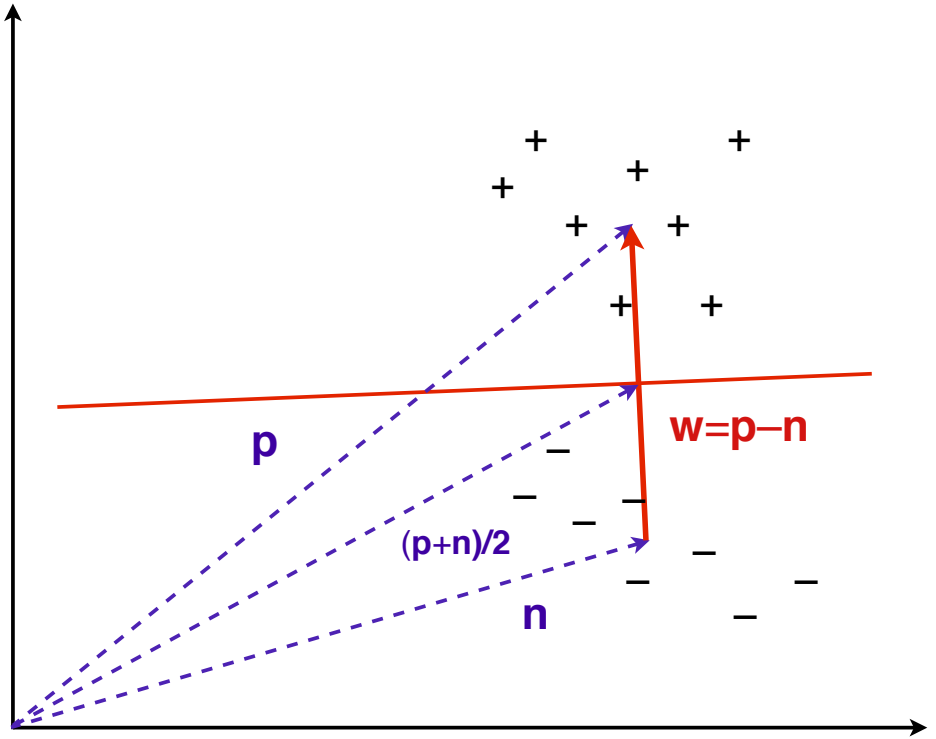
- *Geometric* models use intuitions from geometry such as separating (hyper-)planes, linear transformations and distance metrics.
- *Probabilistic* models view learning as a process of reducing uncertainty, modelled by means of probability distributions.
- *Logical* models are defined in terms of easily interpretable logical expressions.

Alternatively, they can be characterised by their *modus operandi*:

- *Grouping models* divide the instance space into segments; in each segment a very simple (e.g., constant) model is learned.
- *Grading models* learning a single, global model over the instance space.

Figure 1.1, p.22

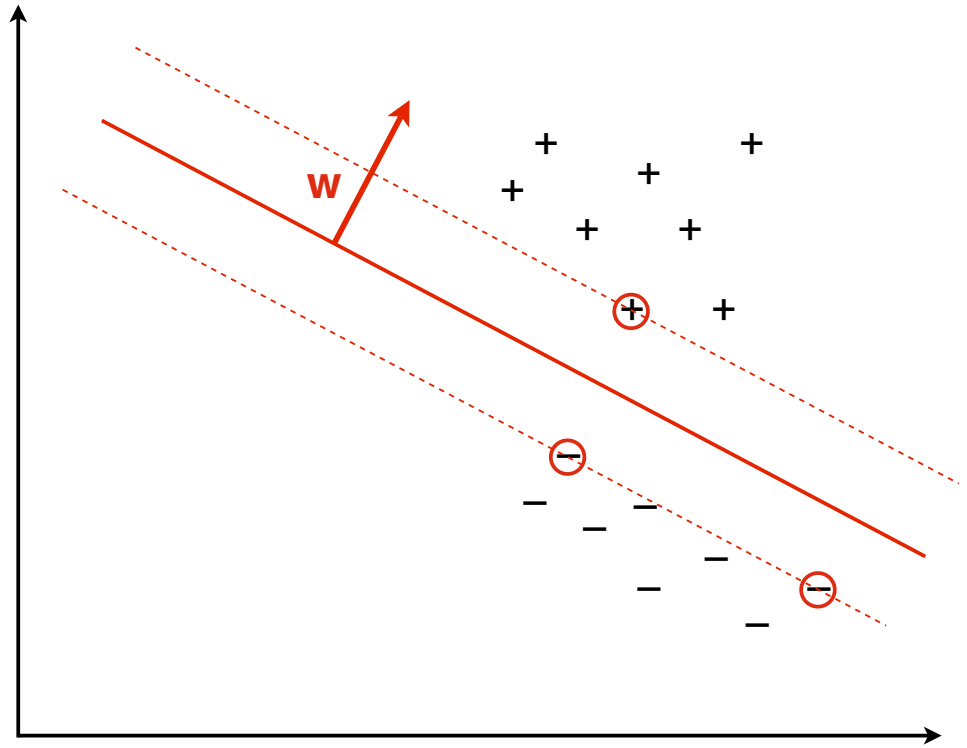
Basic linear classifier



The basic linear classifier constructs a decision boundary by half-way intersecting the line between the positive and negative centres of mass. It is described by the equation $\mathbf{w} \cdot \mathbf{x} = t$, with $\mathbf{w} = \mathbf{p} - \mathbf{n}$; the decision threshold can be found by noting that $(\mathbf{p} + \mathbf{n})/2$ is on the decision boundary, and hence $t = (\mathbf{p} - \mathbf{n}) \cdot (\mathbf{p} + \mathbf{n})/2 = (||\mathbf{p}||^2 - ||\mathbf{n}||^2)/2$, where $||\mathbf{x}||$ denotes the length of vector $\mathbf{x}$.

Figure 1.2, p.23

Support vector machine



The decision boundary learned by a support vector machine from the linearly separable data from Figure 1. The decision boundary maximises the margin, which is indicated by the dotted lines. The circled data points are the support vectors.

Table 1.2, p.26

A simple probabilistic model

Viagra	lottery	$P(Y = \text{spam}\|\text{Viagra}, \text{lottery})$	$P(Y = \text{ham}\|\text{Viagra}, \text{lottery})$
0	0	0.31	**0.69**
0	1	**0.65**	0.35
1	0	**0.80**	0.20
1	1	0.40	**0.60**

'Viagra' and 'lottery' are two Boolean features; Y is the class variable, with values 'spam' and 'ham'. In each row the most likely class is indicated in bold.

Decision rule

Assuming that X and Y are the only variables we know and care about, the posterior distribution $P(Y|X)$ helps us to answer many questions of interest.

- For instance, to classify a new e-mail we determine whether the words 'Viagra' and 'lottery' occur in it, look up the corresponding probability $P(Y = \text{spam}|\text{Viagra}, \text{lottery})$, and predict spam if this probability exceeds 0.5 and ham otherwise.
- Such a recipe to predict a value of Y on the basis of the values of X and the posterior distribution $P(Y|X)$ is called a *decision rule*.

Likelihood ratio

As a matter of fact, statisticians work very often with different conditional probabilities, given by the *likelihood function* $P(X|Y)$.

- I like to think of these as thought experiments: if somebody were to send me a spam e-mail, how likely would it be that it contains exactly the words of the e-mail I'm looking at? And how likely if it were a ham e-mail instead?
- What really matters is not the magnitude of these likelihoods, but their ratio: how much more likely is it to observe this combination of words in a spam e-mail than it is in a non-spam e-mail.
- For instance, suppose that for a particular e-mail described by X we have $P(X|Y = \text{spam}) = 3.5 \cdot 10^{-5}$ and $P(X|Y = \text{ham}) = 7.4 \cdot 10^{-6}$, then observing X in a spam e-mail is nearly five times more likely than it is in a ham e-mail.
- This suggests the following decision rule: predict spam if the likelihood ratio is larger than 1 and ham otherwise.

Important point to remember

Use likelihoods if you want to ignore the prior distribution or assume it uniform, and posterior probabilities otherwise.

Example 1.3, p.28

Posterior odds

$$\frac{P(Y = \text{spam}|\text{Viagra} = 0, \text{lottery} = 0)}{P(Y = \text{ham}|\text{Viagra} = 0, \text{lottery} = 0)} = \frac{0.31}{0.69} = 0.45$$
$$\frac{P(Y = \text{spam}|\text{Viagra} = 1, \text{lottery} = 1)}{P(Y = \text{ham}|\text{Viagra} = 1, \text{lottery} = 1)} = \frac{0.40}{0.60} = 0.67$$
$$\frac{P(Y = \text{spam}|\text{Viagra} = 0, \text{lottery} = 1)}{P(Y = \text{ham}|\text{Viagra} = 0, \text{lottery} = 1)} = \frac{0.65}{0.35} = 1.9$$
$$\frac{P(Y = \text{spam}|\text{Viagra} = 1, \text{lottery} = 0)}{P(Y = \text{ham}|\text{Viagra} = 1, \text{lottery} = 0)} = \frac{0.80}{0.20} = 4.0$$

Using a MAP decision rule we predict ham in the top two cases and spam in the bottom two. Given that the full posterior distribution is all there is to know about the domain in a statistical sense, these predictions are the best we can do: they are *Bayes-optimal*.

Table 1.3, p.29

Example marginal likelihoods

Y	$P(\text{Viagra} = 1 \mid Y)$	$P(\text{Viagra} = 0 \mid Y)$
spam	0.40	0.60
ham	0.12	0.88

Y	$P(\text{lottery} = 1 \mid Y)$	$P(\text{lottery} = 0 \mid Y)$
spam	0.21	0.79
ham	0.13	0.87

Example 1.4, p.30

Using marginal likelihoods

Using the marginal likelihoods from Table 1.3, we can approximate the likelihood ratios (the previously calculated odds from the full posterior distribution are shown in brackets):

$$\frac{P(\text{Viagra}=0|Y=\text{spam})}{P(\text{Viagra}=0|Y=\text{ham})}\frac{P(\text{lottery}=0|Y=\text{spam})}{P(\text{lottery}=0|Y=\text{ham})}=\frac{0.60}{0.88}\frac{0.79}{0.87}=0.62 \quad (0.45)$$

$$\frac{P(\text{Viagra}=0|Y=\text{spam})}{P(\text{Viagra}=0|Y=\text{ham})}\frac{P(\text{lottery}=1|Y=\text{spam})}{P(\text{lottery}=1|Y=\text{ham})}=\frac{0.60}{0.88}\frac{0.21}{0.13}=1.1 \quad (1.9)$$

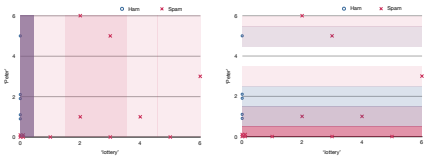
$$\frac{P(\text{Viagra}=1|Y=\text{spam})}{P(\text{Viagra}=1|Y=\text{ham})}\frac{P(\text{lottery}=0|Y=\text{spam})}{P(\text{lottery}=0|Y=\text{ham})}=\frac{0.40}{0.12}\frac{0.79}{0.87}=3.0 \quad (4.0)$$

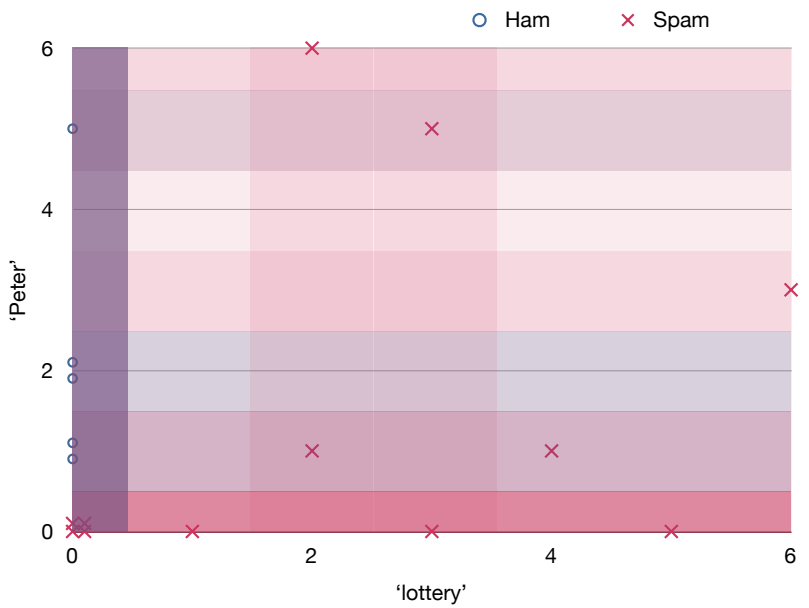
$$\frac{P(\text{Viagra}=1|Y=\text{spam})}{P(\text{Viagra}=1|Y=\text{ham})}\frac{P(\text{lottery}=1|Y=\text{spam})}{P(\text{lottery}=1|Y=\text{ham})}=\frac{0.40}{0.12}\frac{0.21}{0.13}=5.4 \quad (0.67)$$

We see that, using a maximum likelihood decision rule, our very simple model arrives at the Bayes-optimal prediction in the first three cases, but not in the fourth ('Viagra' and 'lottery' both present), where the marginal likelihoods are actually very misleading.

Figure 1.3, p.31

The Scottish classifier

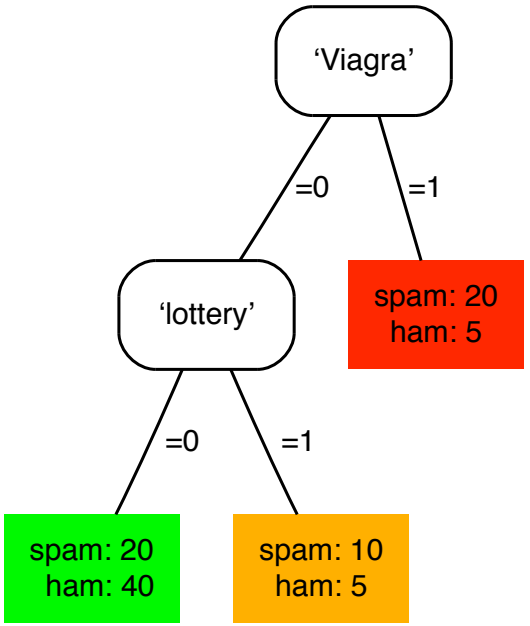


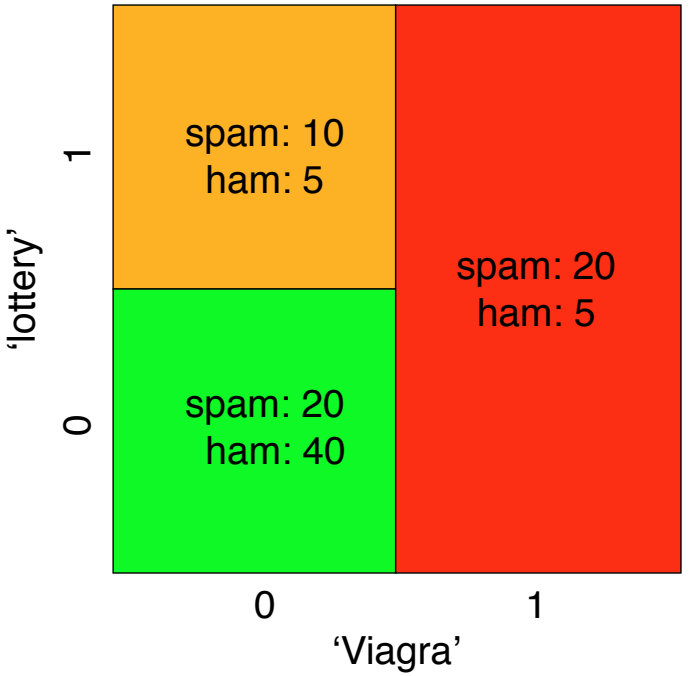


(top) Visualisation of two marginal likelihoods as estimated from a small data set. The colours indicate whether the likelihood points to spam or ham. **(bottom)** Combining the two marginal likelihoods gives a pattern not unlike that of a Scottish tartan.

Figure 1.4, p.32

A feature tree





(left) A feature tree combining two Boolean features. Each internal node or split is labelled with a feature, and each edge emanating from a split is labelled with a feature value. Each leaf therefore corresponds to a unique combination of feature values. Also indicated in each leaf is the class distribution derived from the training set. **(right)** A feature tree partitions the instance space into rectangular regions, one for each leaf. We can clearly see that the majority of ham lives in the lower left-hand corner.

Example 1.5, p.33

Labelling a feature tree

- The leaves of the tree in Figure 1.4 could be labelled, from left to right, as ham – spam – spam, employing a simple decision rule called *majority class*.
- Alternatively, we could label them with the proportion of spam e-mail occurring in each leaf: from left to right, 1/3, 2/3, and 4/5.
- Or, if our task was a regression task, we could label the leaves with predicted real values or even linear functions of some other, real-valued features.

What's next?

1. The ingredients of machine learning
   - Tasks: the problems that can be solved with machine learning
     - Looking for structure
   - Models: the output of machine learning
     - Geometric models
     - Probabilistic models
     - Logical models
   - Features: the workhorses of machine learning
     - Two uses of features
     - Feature construction and transformation

Example 1.7, p.39

The MLM data set

Suppose we have a number of learning models that we want to describe in terms of a number of properties:

- the extent to which the models are geometric, probabilistic or logical;
- whether they are grouping or grading models;
- the extent to which they can handle discrete and/or real-valued features;
- whether they are used in supervised or unsupervised learning; and
- the extent to which they can handle multi-class problems.

The first two properties could be expressed by discrete features with three and two values, respectively; or if the distinctions are more gradual, each aspect could be rated on some numerical scale. A simple approach would be to measure each property on an integer scale from 0 to 3, as in Table 1.4. This table establishes a data set in which each row represents an instance and each column a feature.

Table 1.4, p.39

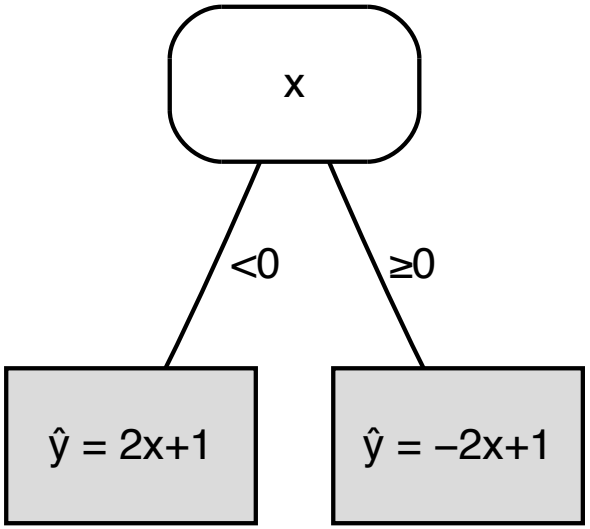
The MLM data set

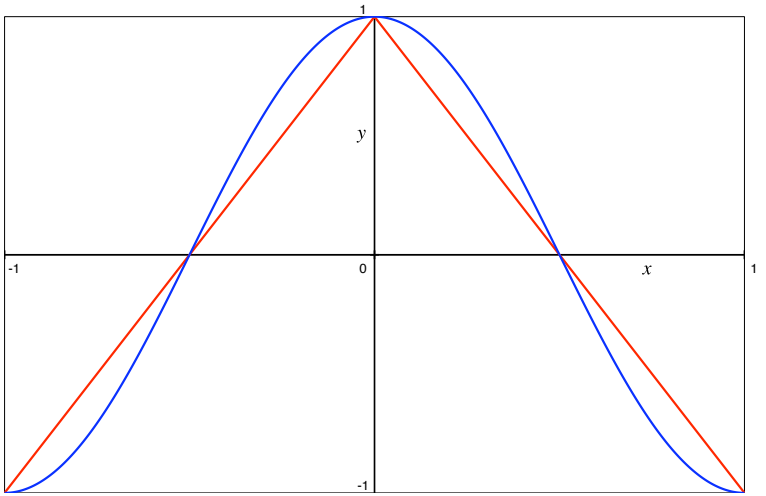
Model	geom	stats	logic	group	grad	disc	real	sup	unsup	multi
Trees	1	0	3	3	0	3	2	3	2	3
Rules	0	0	3	3	1	3	2	3	0	2
naive Bayes	1	3	1	3	1	3	1	3	0	3
kNN	3	1	0	2	2	1	3	3	0	3
Linear Classifier	3	0	0	0	3	1	3	3	0	0
Linear Regression	3	1	0	0	3	0	3	3	0	1
Logistic Regression	3	2	0	0	3	1	3	3	0	0
SVM	2	2	0	0	3	2	3	3	0	0
Kmeans	3	2	0	1	2	1	3	0	3	1
GMM	1	3	0	0	3	1	3	0	3	1
Associations	0	0	3	3	0	3	1	0	3	1

The MLM data set describing properties of machine learning models.

Figure 1.9, p.41

A small regression tree

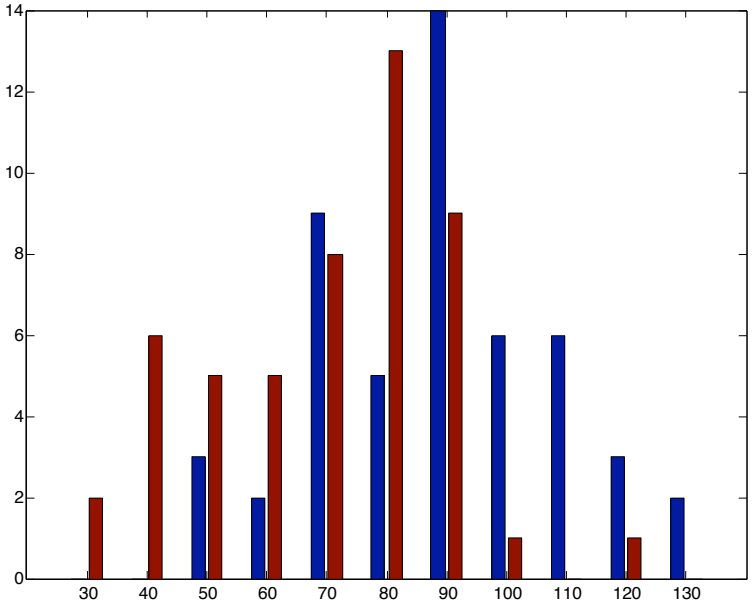


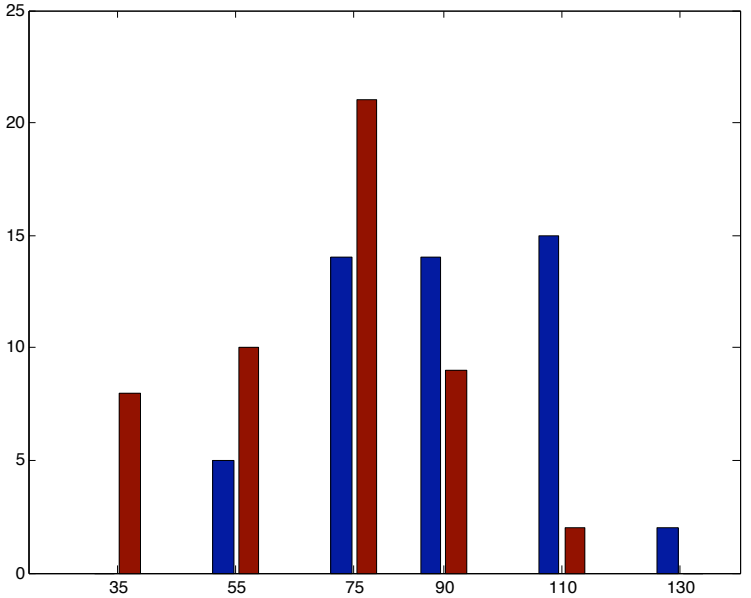


(left) A regression tree combining a one-split feature tree with linear regression models in the leaves. Notice how x is used as both a splitting feature and a regression variable. **(right)** The function $y = \cos \pi x$ on the interval $-1 \leq x \leq 1$, and the piecewise linear approximation achieved by the regression tree.

Figure 1.10, p.42

Class-sensitive discretisation





(left) Artificial data depicting a histogram of body weight measurements of people with (blue) and without (red) diabetes, with eleven fixed intervals of 10 kilograms width each. **(right)** By joining the first and second, third and fourth, fifth and sixth, and the eighth, ninth and tenth intervals, we obtain a discretisation such that the proportion of diabetes cases increases from left to right. This discretisation makes the feature more useful in predicting diabetes.

Example 1.9, p.43

The kernel trick

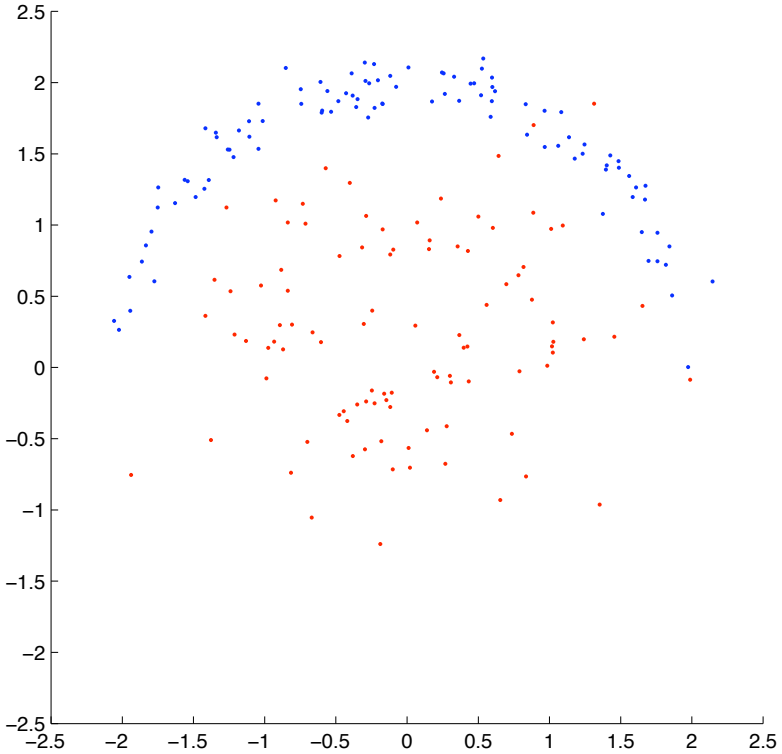
Let $\mathbf{x}_1 = (x_1, y_1)$ and $\mathbf{x}_2 = (x_2, y_2)$ be two data points, and consider the mapping $(x, y) \mapsto (x^2, y^2, \sqrt{2}xy)$ to a three-dimensional feature space. The points in feature space corresponding to $\mathbf{x}_1$ and $\mathbf{x}_2$ are $\mathbf{x}'_1 = (x_1^2, y_1^2, \sqrt{2}x_1 y_1)$ and $\mathbf{x}'_2 = (x_2^2, y_2^2, \sqrt{2}x_2 y_2)$. The dot product of these two feature vectors is

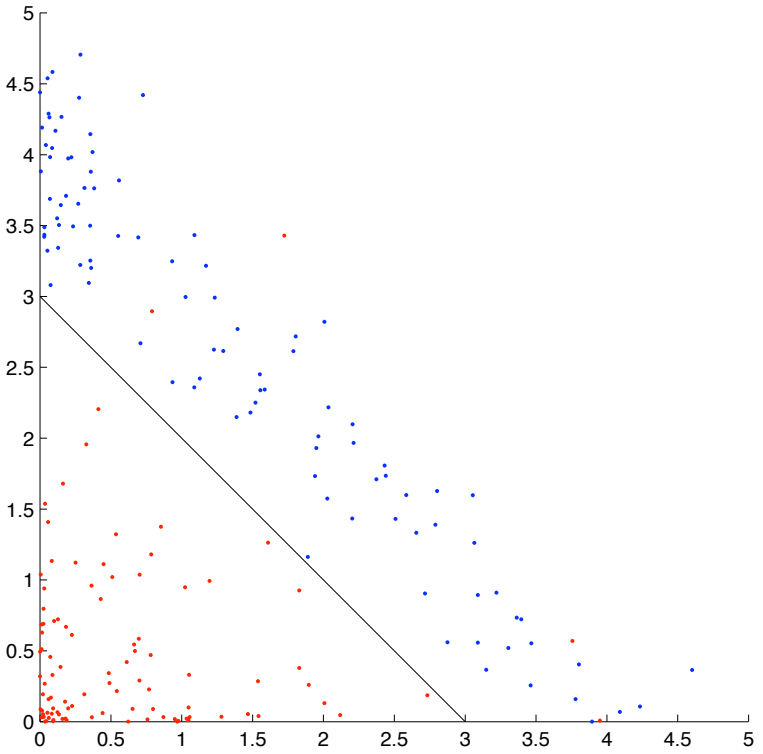
$$\mathbf{x}'_1 \cdot \mathbf{x}'_2 = x_1^2 x_2^2 + y_1^2 y_2^2 + 2x_1 y_1 x_2 y_2 = (x_1 x_2 + y_1 y_2)^2 = (\mathbf{x}_1 \cdot \mathbf{x}_2)^2$$

That is, by squaring the dot product in the original space we obtain the dot product in the new space *without actually constructing the feature vectors*! A function that calculates the dot product in feature space directly from the vectors in the original space is called a *kernel* – here the kernel is $\kappa(\mathbf{x}_1, \mathbf{x}_2) = (\mathbf{x}_1 \cdot \mathbf{x}_2)^2$.

Figure 1.11, p.43

Non-linearly separable data





(left) A linear classifier would perform poorly on this data. **(right)** By transforming the original (x, y) data into $(x', y') = (x^2, y^2)$, the data becomes more 'linear', and a linear decision boundary $x' + y' = 3$ separates the data fairly well. In the original space this corresponds to a circle with radius $\sqrt{3}$ around the origin.