Demystifying Machine Learning

A Jean Golding Institute workshop

Peter Flach and Niall Twomey

Intelligent Systems Laboratory, University of Bristol, United Kingdom

December 5, 2017

Part One

About today

2.00 – 3.00pm A closer look in a machine learning practitioner's toolbox (Peter)

3.00 – 3.30pm Discussion over coffee

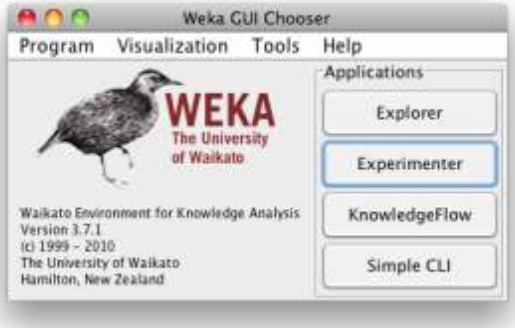
3.30 – 4.30pm Machine learning and data science in practice (Niall)

What we won't cover today:

- deep learning: perhaps a JGI workshop in Spring
- refinforcement learning: learning by trial and error
- artificial intelligence: How did Watson manage to win at Jeopardy? Or AlphaGo at Go? This is part of a much bigger story involving natural language understanding, search, planning, etc.

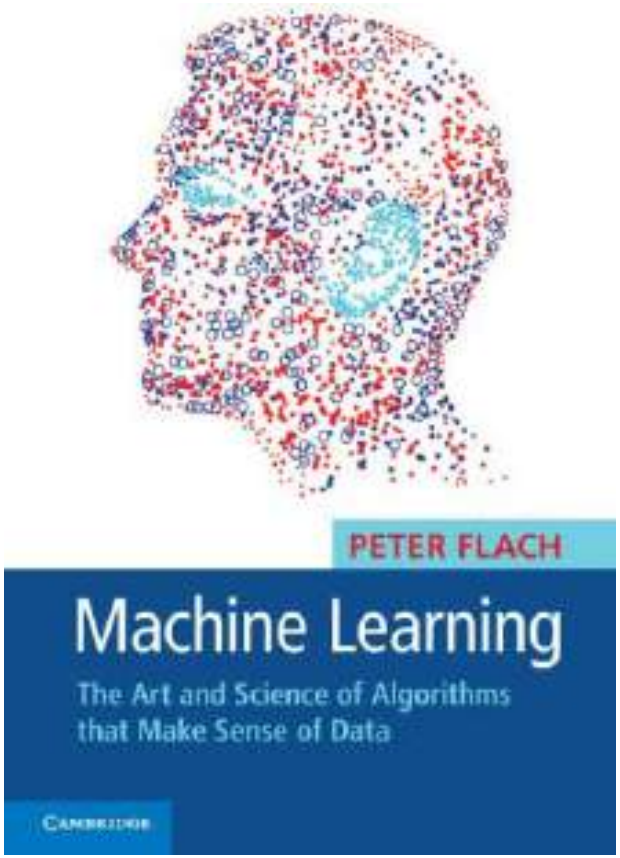
Demos

As far as time allows Peter will include some demos using Weka:
www.cs.waikato.ac.nz/ml/weka/downloading.html



After the break, Niall will demo data wrangling etc. using Jupyter notebooks (Python).

Shameless plug...



Peter's presentation consists of a selection of slides accompanying the above book published by Cambridge University Press in 2012.

Go to the URL in the footer for more.

Table of contents

Table of contents II

Table of contents III

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.

```
-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
```

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.

Table 1, p.3

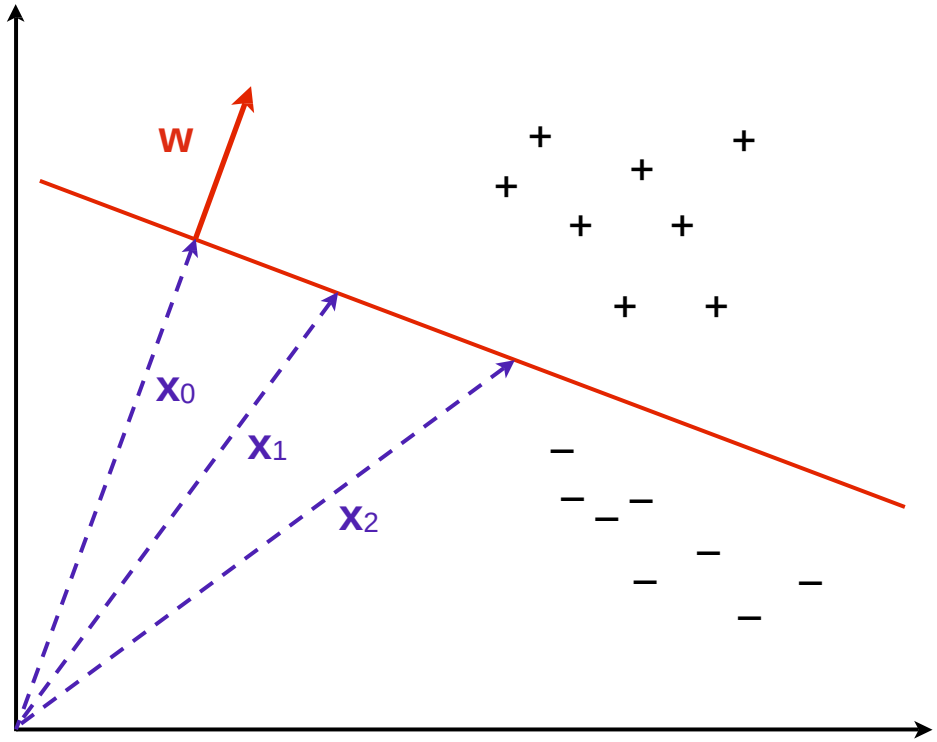
Spam filtering as a classification task

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



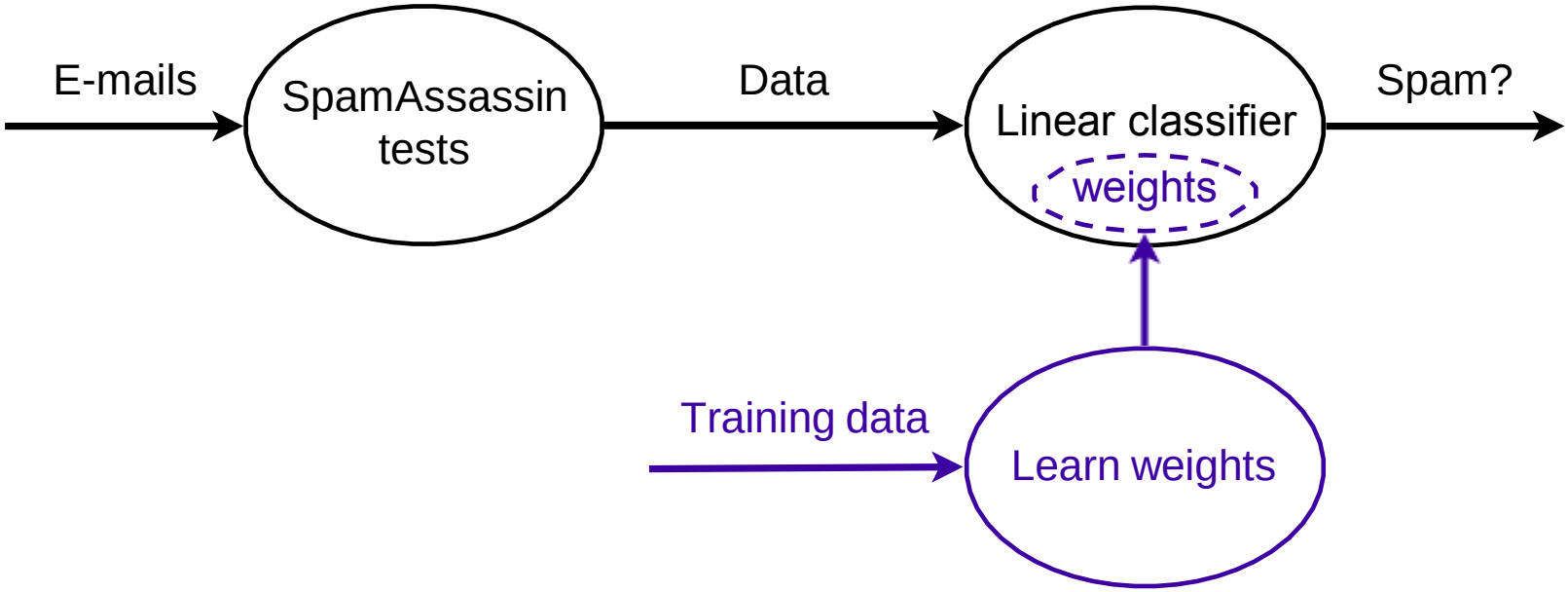
The straight line separates the positives from the negatives. It is defined by $\mathbf{w} \cdot \mathbf{x}_i = t$, where $\mathbf{w}$ is a vector perpendicular to the decision boundary and pointing in the direction of the positives, t is the decision threshold, and $\mathbf{x}_i$ points to a point on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

A definition of machine learning

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

Figure 2, p.5

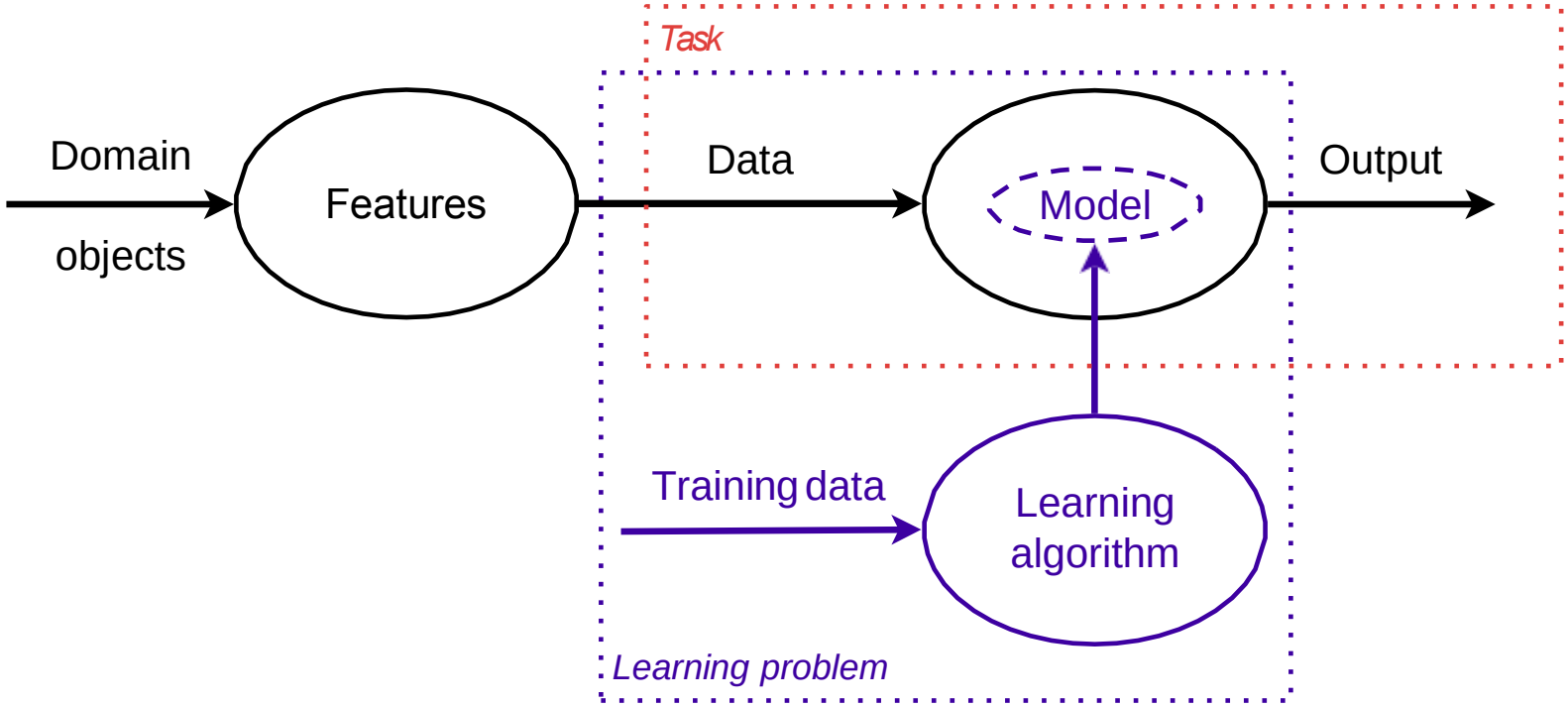
Machine learning for spam filtering



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 (inblue) we see the bit that is done by machine learning.

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).

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.

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.

Machine learning models

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

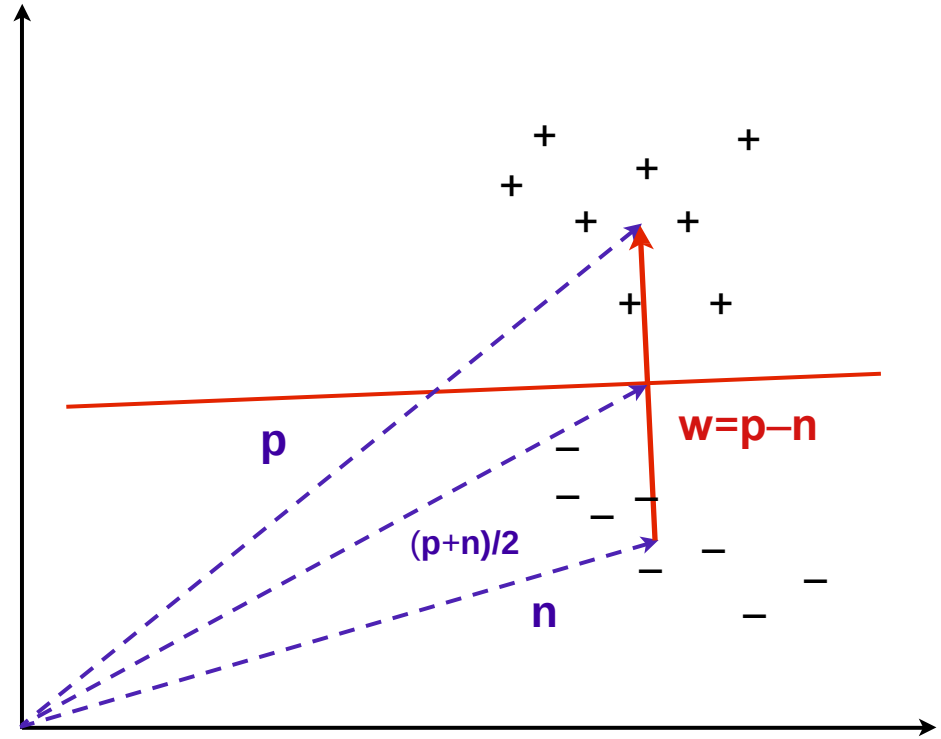
- *Geometric* models use intuitions from geometry such as separating (hyper-)planes, linear transformations and distance metrics.
- *Probabilistic* models view learning as a process of reducing uncertainty, modelled by means of probability distributions.
- *Logical* models are defined in terms of easily 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* learn a single, global model over the instance space.

Figure 1.1, p.22

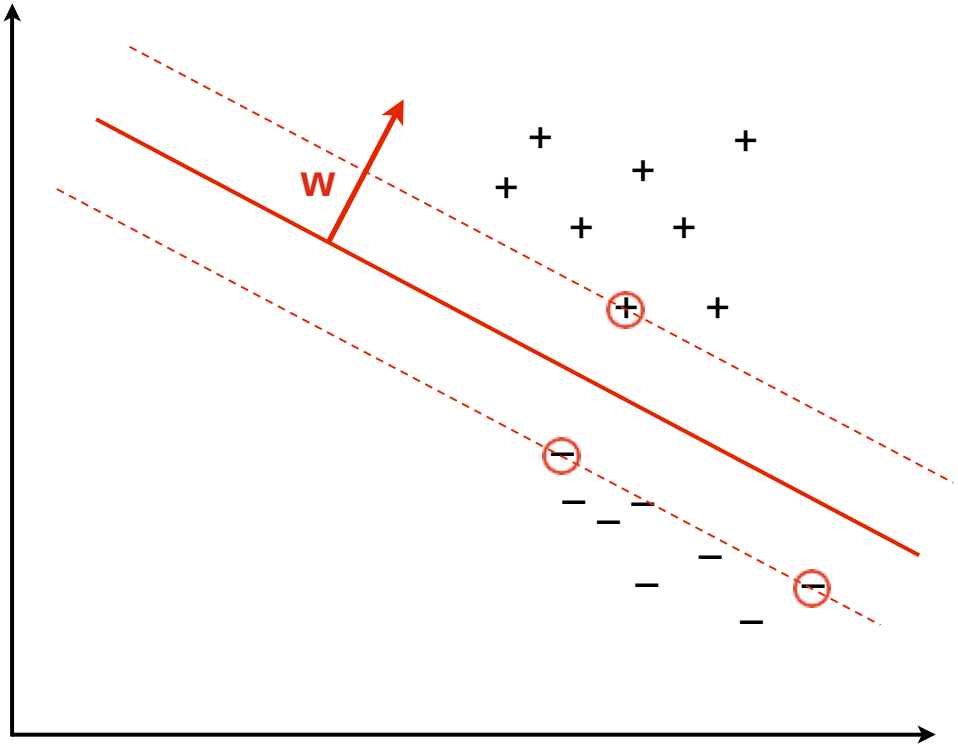
Basic linear classifier



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 fromFigure 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

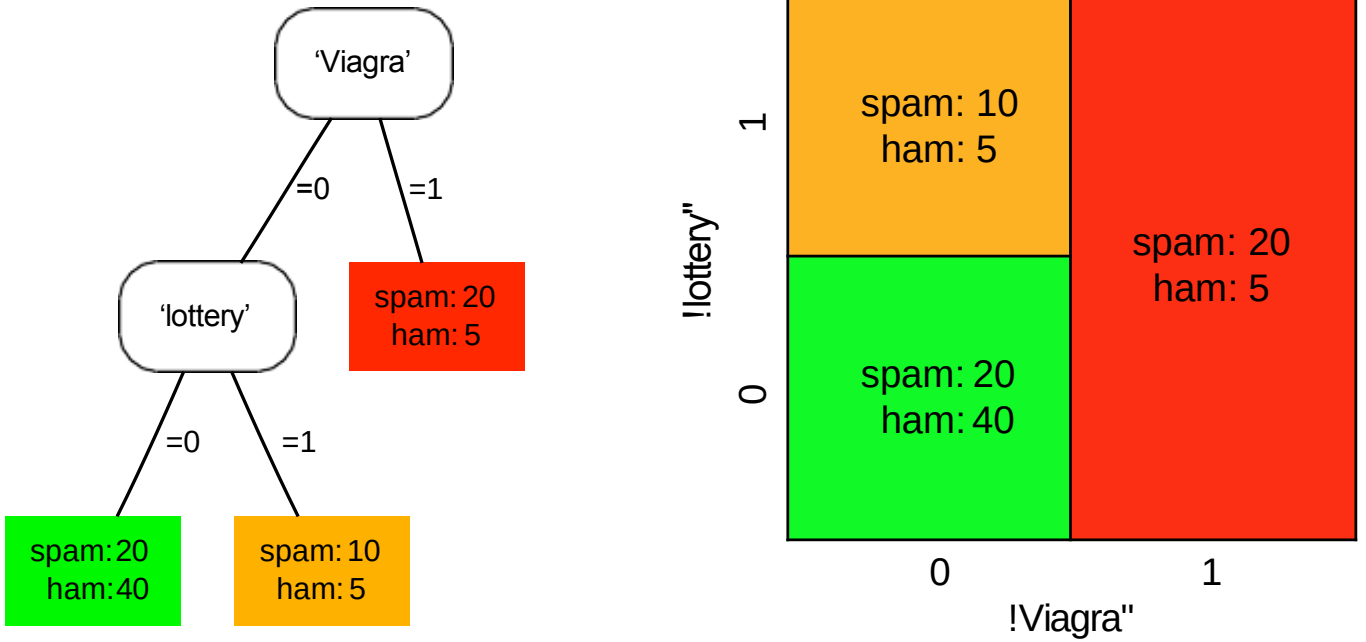
A simple probabilistic model

Viagra	lottery	$P(Y = \text{spam} \mid \text{Viagra, lottery})$	$P(Y = \text{ham} \mid \text{Viagra, lottery})$
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.

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

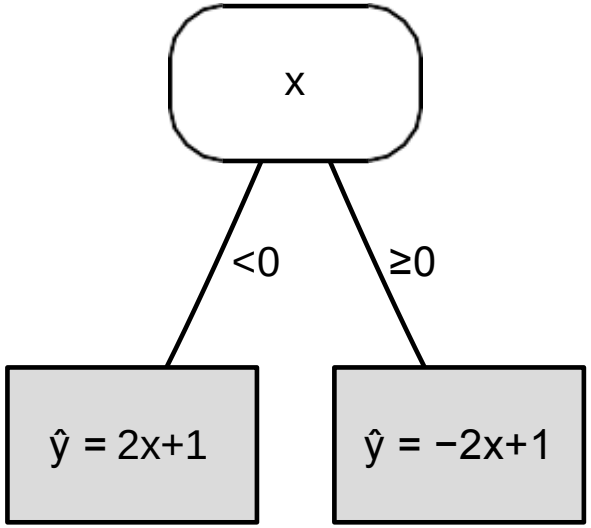
Labelling a feature tree

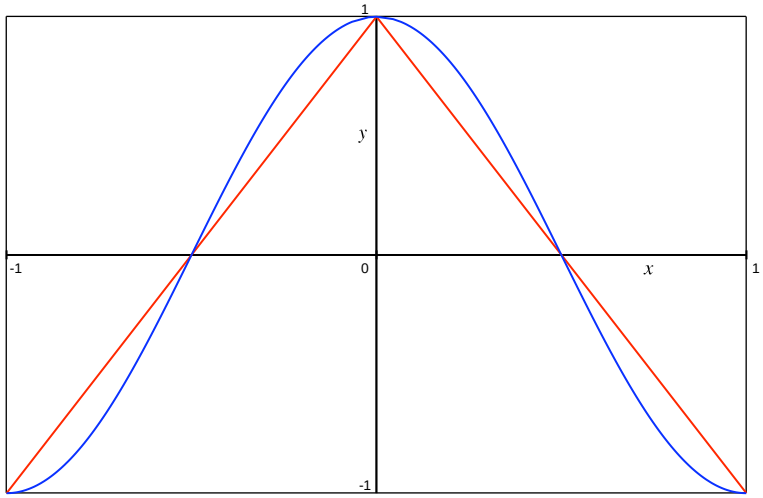
Trees can be used to predict classes, probabilities, numbers, even functions.

- 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.

Figure 1.9, p.41

A small regression tree



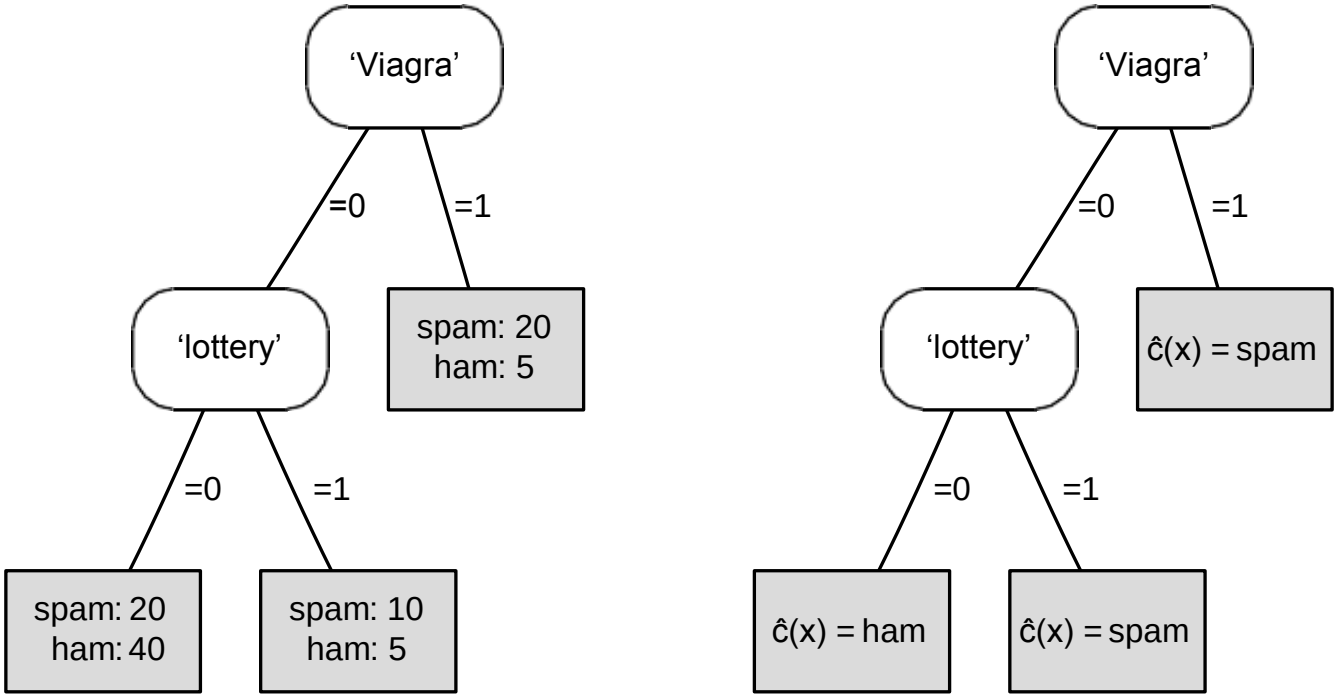


(left) A regression tree combining a one-split feature tree with linear regression models in the leaves. Notice how x is used as both a splitting feature and a regression variable. **(right)** The function $y = \cos \pi x$ on the interval $-1 \leq x \leq 1$, and the piecewise linear approximation achieved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

What's next?

Figure 2.1, p.53

A decision tree



(left) A feature tree with training set class distribution in the leaves. **(right)** A decision tree obtained using the majority class decision rule.

Table 2.2, p.54

Contingency table

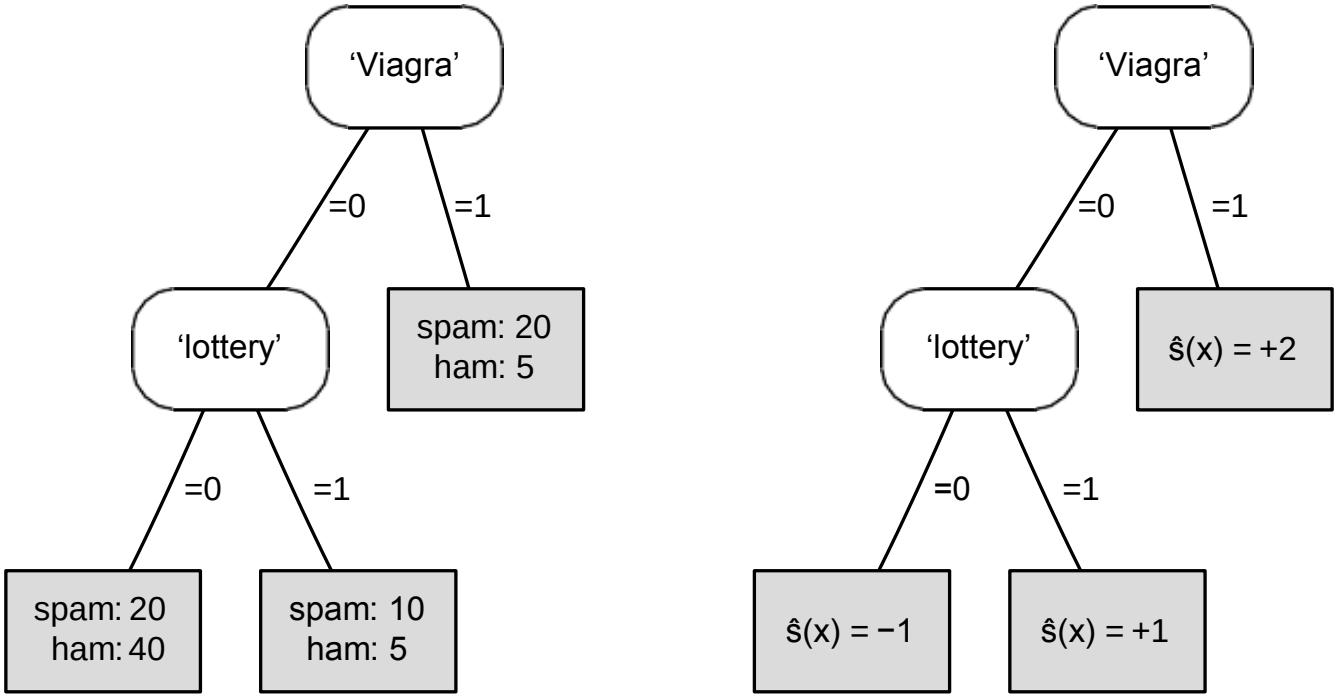
	Predicted ⊕	*Predicted* 8	
Actual ⊕	**30**	**20**	50
Actual 8	**10**	**40**	50
	40	60	100

	⊕	8	
⊕	**20**	**30**	50
8	**20**	**30**	50
	40	60	100

(left) A two-class contingency table or confusion matrix depicting the performance of the decision tree inFigure 2.1. Numbers on the descending diagonal indicate correct predictions, while the ascending diagonal concerns prediction errors. **(right)** A contingency table with the same marginals but independent rows and columns.

Figure 2.5, p.62

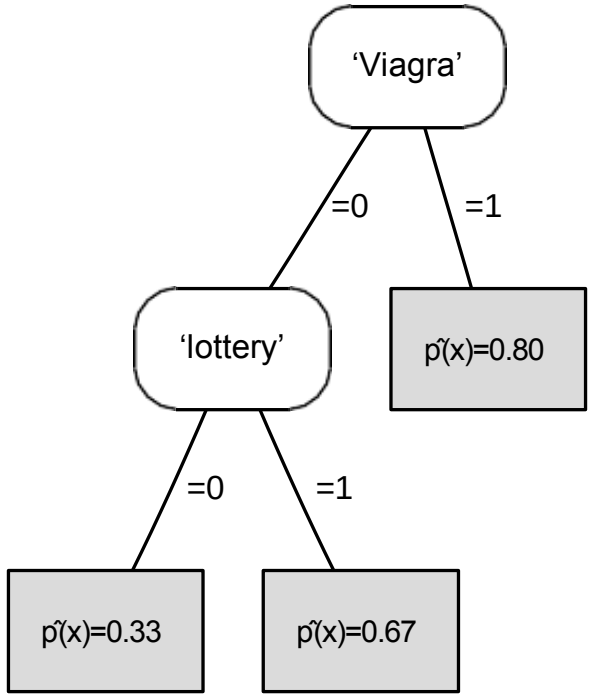
A scoring tree



(left) A feature tree with training set class distribution in the leaves. **(right)** A scoring tree using the logarithm of the class ratio as scores; spam is taken as the positive class.

Figure 2.12, p.73

Probability estimation tree



A probability estimation tree derived from the feature tree inFigure 1.4.

What's next?

Example 3.1, p.82

Performance of multi-class classifiers

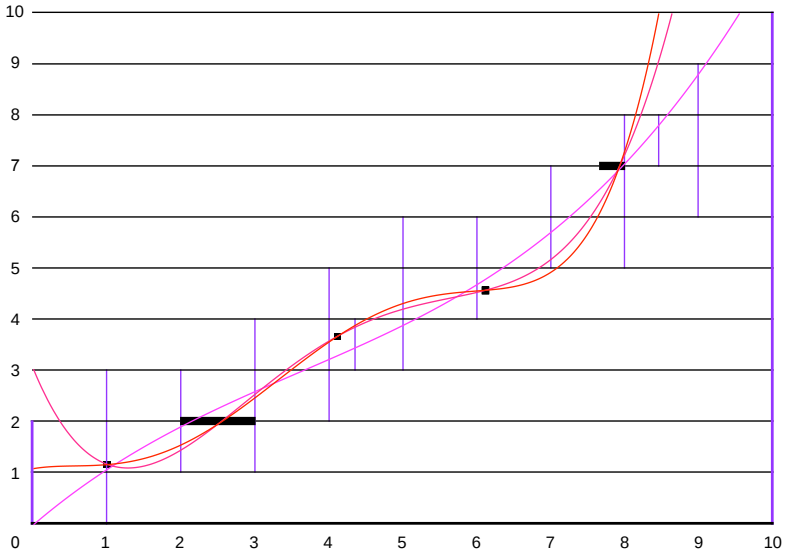
Consider the following three-class confusion matrix (plus marginals):

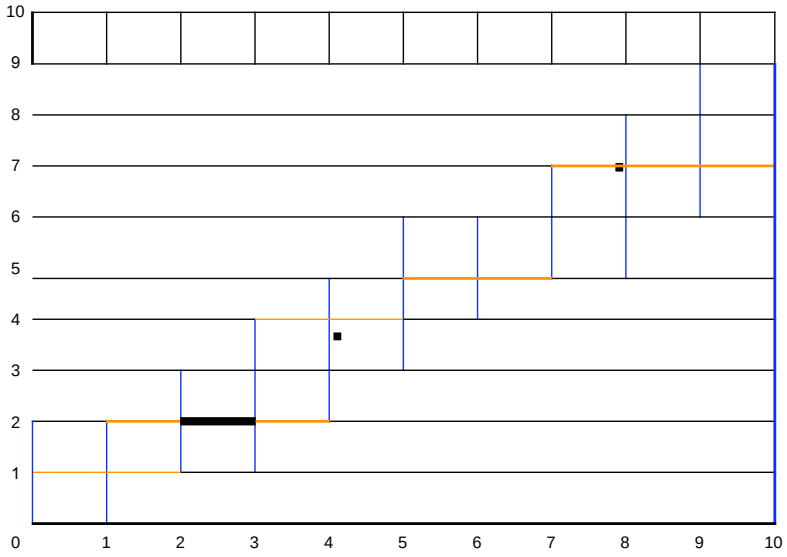
		Predicted			
		15	**23**		20
Actual		**7**	**15**		30
		2	**3**	**45**	50
		24	20	56	100

- The accuracy of this classifier is $(15+15+45)/100 = 0.75$.
- We can calculate per-class precision and recall: for the first class this is $15/24 = 0.63$ and $15/20 = 0.75$ respectively, for the second class $15/20 = 0.75$ and $15/30 = 0.50$, and for the third class $45/56 = 0.80$ and $45/50 = 0.90$.
- We could average these numbers to obtain single precision and recall numbers for the whole classifier, or we could take a weighted average taking the proportion of each class into account. For instance, the weighted average precision is $0.20 \cdot 0.63 + 0.30 \cdot 0.75 + 0.50 \cdot 0.80 = 0.75$.

Figure 3.2, p.92

Fitting polynomials to data





(left) Polynomials of different degree fitted to a set of five points. From bottom to top in the top right-hand corner: degree 1 (straight line), degree 2 (parabola), degree 3, degree 4 (which is the lowest degree able to fit the points exactly), degree 5. **(right)** A piecewise constant function learned by a grouping model; the dotted reference line is the linear function from the left figure.

Example 3.11, p.100

Subgroup discovery

Imagine you want to market the new version of a successful product. You have a database of people who have been sent information about the previous version, containing all kinds of demographic, economic and social information about those people, as well as whether or not they purchased the product.

- If you were to build a classifier or ranker to find the most likely customers for your product, it is unlikely to outperform the majority class classifier (typically, relatively few people will have bought the product).
- However, what you are really interested in is finding reasonably sized subsets of people with a proportion of customers that is significantly higher than in the overall population. You can then target those people in your marketing campaign, ignoring the rest of your database.

Example 3.12, p.101

Association rule discovery

Associations are things that usually occur together. For example, in market basket analysis we are interested in items frequently bought together. An example of an association rule is ‘**if** beer **then** crisps’, stating that customers who buy beer tend to also buy crisps.

- In a motorway service station most clients will buy petrol. This means that there will be many frequent item sets involving petrol, such as {newspaper, petrol}.
- This might suggest the construction of an association rule ‘**if** newspaper **then** petrol’ – however, this is predictable given that {petrol} is already a frequent item set (and clearly at least as frequent as {newspaper, petrol}).
- Of more interest would be the converse rule ‘**if** petrol **then** newspaper’ which expresses that a considerable proportion of the people buying petrol also buy a newspaper.

What's next?

Example 4.1, p.106

Learning conjunctive concepts

Suppose you come across a number of sea animals that you suspect belong to the same species. You observe their length in metres, whether they have gills, whether they have a prominent beak, and whether they have few or many teeth. Using these features, the first animal can described by the following conjunction:

$$\text{Length} = 3 \wedge \text{Gills} = \text{no} \wedge \text{Beak} = \text{yes} \wedge \text{Teeth} = \text{many}$$

The next one has the same characteristics but is a metre longer, so you drop the length condition and generalise the conjunction to

$$\text{Gills} = \text{no} \wedge \text{Beak} = \text{yes} \wedge \text{Teeth} = \text{many}$$

The third animal is again 3 metres long, has a beak, no gills and few teeth, so your description becomes

$$\text{Gills} = \text{no} \wedge \text{Beak} = \text{yes}$$

All remaining animals satisfy this conjunction, and you finally decide they are some kind of dolphin.

Example 4.2, p.110

Negative examples

In Example 4.1 we observed the following dolphins:

p1: $\text{Length} = 3 \wedge \text{Gills} = \text{no} \wedge \text{Beak} = \text{yes} \wedge \text{Teeth} = \text{many}$

p2: $\text{Length} = 4 \wedge \text{Gills} = \text{no} \wedge \text{Beak} = \text{yes} \wedge \text{Teeth} = \text{many}$

p3: $\text{Length} = 3 \wedge \text{Gills} = \text{no} \wedge \text{Beak} = \text{yes} \wedge \text{Teeth} = \text{few}$

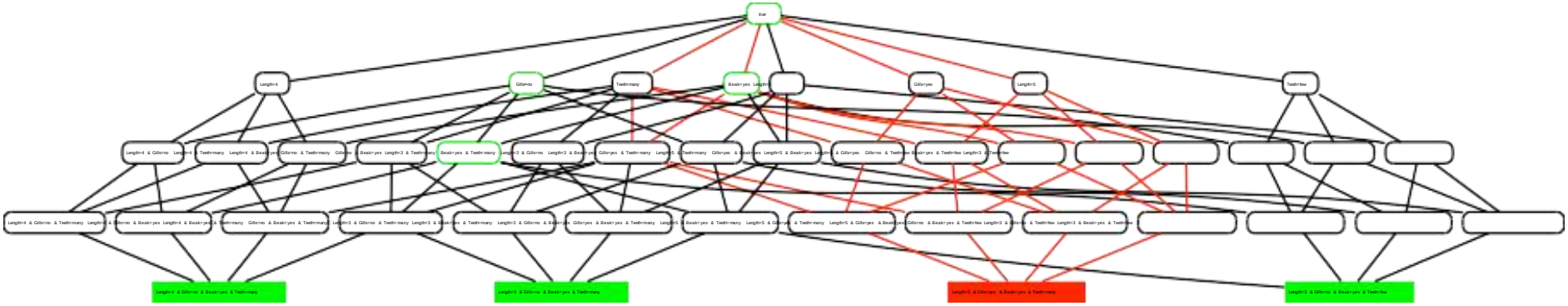
Suppose you next observe an animal that clearly doesn't belong to the species – a negative example. It is described by the following conjunction:

n1: $\text{Length} = 5 \wedge \text{Gills} = \text{yes} \wedge \text{Beak} = \text{yes} \wedge \text{Teeth} = \text{many}$

This negative example rules out some of the generalisations that were hitherto still possible: in particular, it rules out the concept $\text{Beak} = \text{yes}$, as well as the empty concept which postulates that everything is a dolphin.

Figure 4.3, p.111

Employing negative examples



A negative example can rule out some of the generalisations of the LGG of the positive examples. Every concept which is connected by a red path to a negative example covers that negative and is therefore ruled out as a hypothesis. Only two conjunctions cover all positives and no negatives: $\text{Gills} = \text{no} \wedge \text{Beak} = \text{yes}$ and $\text{Gills} = \text{no}$.

What's next?

Algorithm 5.1, p.132

Growing a feature tree

Algorithm GrowTree(D, F)– grow a feature tree from training data.

Input : data D; set of features F.

Output : feature tree T with labelled leaves.

```
if Homogeneous(D) then return Label(D);
S ← BestSplit(D, F);                    //e.g., BestSplit-Class (Algorithm 5.2)
split D into subsets D_i according to the literals in S;
for each i do
    if D_i ≠ ∅ then T_i ← GrowTree(D_i, F);
    else T_i is a leaf labelled with Label(D);
end
return a tree whose root is labelled with S and whose children are T_i
```

Growing a feature tree

Algorithm 5.1 gives the generic learning procedure common to most tree learners. It assumes that the following three functions are defined:

Homogeneous(D) returns true if the instances in D are homogeneous enough to be labelled with a single label, and false otherwise;

Label(D) returns the most appropriate label for a set of instances D;

BestSplit(D, F) returns the best set of literals to be put at the root of the tree.

These functions depend on the task at hand: for instance, for classification tasks a set of instances is homogeneous if they are (mostly) of a single class, and the most appropriate label would be the majority class. For clustering tasks a set of instances is homogenous if they are close together, and the most appropriate label would be some exemplar such as the mean.

What's next?

Example 6.1, p.159

Learning a rule list

Consider again our small dolphins data set with positive examples

p1: $\text{Length} = 3 \wedge \text{Gills} = \text{no} \wedge \text{Beak} = \text{yes} \wedge \text{Teeth} = \text{many}$

p2: $\text{Length} = 4 \wedge \text{Gills} = \text{no} \wedge \text{Beak} = \text{yes} \wedge \text{Teeth} = \text{many}$

p3: $\text{Length} = 3 \wedge \text{Gills} = \text{no} \wedge \text{Beak} = \text{yes} \wedge \text{Teeth} = \text{few}$

p4: $\text{Length} = 5 \wedge \text{Gills} = \text{no} \wedge \text{Beak} = \text{yes} \wedge \text{Teeth} = \text{many}$

p5: $\text{Length} = 5 \wedge \text{Gills} = \text{no} \wedge \text{Beak} = \text{yes} \wedge \text{Teeth} = \text{few}$

and negatives

n1: $\text{Length} = 5 \wedge \text{Gills} = \text{yes} \wedge \text{Beak} = \text{yes} \wedge \text{Teeth} = \text{many}$

n2: $\text{Length} = 4 \wedge \text{Gills} = \text{yes} \wedge \text{Beak} = \text{yes} \wedge \text{Teeth} = \text{many}$

n3: $\text{Length} = 5 \wedge \text{Gills} = \text{yes} \wedge \text{Beak} = \text{no} \wedge \text{Teeth} = \text{many}$

n4: $\text{Length} = 4 \wedge \text{Gills} = \text{yes} \wedge \text{Beak} = \text{no} \wedge \text{Teeth} = \text{many}$

n5: $\text{Length} = 4 \wedge \text{Gills} = \text{no} \wedge \text{Beak} = \text{yes} \wedge \text{Teeth} = \text{few}$