Machine Learning

The Art and Science of Algorithms
that Make Sense of Data

Peter Flach

Excerpts:
Background material
and literature references

There are a number of useful ways in which we can express the SpamAssassin classifier in mathematical notation. If we denote the result of the i-th test for a given e-mail as x_i, where $x_i = 1$ if the test succeeds and 0 otherwise, and we denote the weight of the i-th test as w_i, then the total score of an e-mail can be expressed as $\sum_{i=1}^{n} w_i x_i$, making use of the fact that w_i contributes to the sum only if $x_i = 1$, i.e., if the test succeeds for the e-mail. Using t for the threshold above which an e-mail is classified as spam (5 in our example), the 'decision rule' can be written as $\sum_{i=1}^{n} w_i x_i > t$.

Notice that the left-hand side of this inequality is linear in the x_i variables, which essentially means that increasing one of the x_i by a certain amount, say δ, will change the sum by an amount $(w_i\delta)$ that is independent of the value of x_i. This wouldn't be true if x_i appeared squared in the sum, or with any exponent other than 1.

The notation can be simplified by means of linear algebra, writing $\mathbf{w}$ for the vector of weights $(w_1,\dots,w_n)$ and $\mathbf{x}$ for the vector of test results $(x_1,\dots,x_n)$. The above inequality can then be written using a dot product: $\mathbf{w}\cdot\mathbf{x} > t$. Changing the inequality to an equality $\mathbf{w}\cdot\mathbf{x} = t$, we obtain the 'decision boundary', separating spam from ham. The decision boundary is a plane (a 'straight' surface) in the space spanned by the x_i variables because of the linearity of the left-hand side. The vector $\mathbf{w}$ is perpendicular to this plane and points in the direction of spam. Figure 1 visualises this for two variables.

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 $\mathbf{x}^\circ = (1, x_1, \dots, x_n)$ and the extended weight vector is $\mathbf{w}^\circ = (-t, w_1, \dots, w_n)$, 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 (but note that this doesn't really affect the data, as all data points and the 'real' decision boundary live in the plane $x_0 = 1$).

Background 1. SpamAssassin in mathematical notation. In boxes such as these, I will briefly remind you of useful concepts and notation. If some of these are unfamiliar, you will need to spend some time reviewing them – using other books or online resources such as `www.wikipedia.org` or `mathworld.wolfram.com` – to fully appreciate the rest of the book.

Probabilities involve ‘random variables’ that describe outcomes of ‘events’. These events are often hypothetical and therefore probabilities have to be estimated. For example, consider the statement ‘42% of the UK population approves of the current Prime Minister’. The only way to know this for certain is to ask everyone in the UK, which is of course unfeasible. Instead, a (hopefully representative) sample is queried, and a more correct statement would then be ‘42% of a sample drawn from the UK population approves of the current Prime Minister’, or ‘the proportion of the UK population approving of the current Prime Minister is estimated at 42%’. Notice that these statements are formulated in terms of proportions or ‘relative frequencies’; a corresponding statement expressed in terms of probabilities would be ‘the probability that a person uniformly drawn from the UK population approves of the current Prime Minister is estimated at 0.42’. The event here is ‘this random person approves of the PM’.

The ‘conditional probability’ $P(A|B)$ is the probability of event A happening given that event B happened. For instance, the approval rate of the Prime Minister may differ for men and women. Writing $P(\text{PM})$ for the probability that a random person approves of the Prime Minister and $P(\text{PM}|\text{woman})$ for the probability that a random woman approves of the Prime Minister, we then have that $P(\text{PM}|\text{woman}) = P(\text{PM},\text{woman})/P(\text{woman})$, where $P(\text{PM},\text{woman})$ is the probability of the ‘joint event’ that a random person both approves of the PM and is a woman, and $P(\text{woman})$ is the probability that a random person is a woman (i.e., the proportion of women in the UK population).

Other useful equations include $P(A,B) = P(A|B)P(B) = P(B|A)P(A)$ and $P(A|B) = P(B|A)P(A)/P(B)$. The latter is known as ‘Bayes’ rule’ and will play an important role in this book. Notice that many of these equations can be extended to more than two random variables, e.g. the ‘chain rule of probability’: $P(A,B,C,D) = P(A|B,C,D)P(B|C,D)P(C|D)P(D)$.

Two events A and B are independent if $P(A|B) = P(A)$, i.e., if knowing that B happened doesn’t change the probability of A happening. An equivalent formulation is $P(A,B) = P(A)P(B)$. In general, multiplying probabilities involves the assumption that the corresponding events are independent.

The ‘odds’ of an event is the ratio of the probability that the event happens and the probability that it doesn’t happen. That is, if the probability of a particular event happening is p, then the corresponding odds are $o = p/(1-p)$. Conversely, we have that $p = o/(o+1)$. So, for example, a probability of 0.8 corresponds to odds of 4:1, the opposite odds of 1:4 give probability 0.2, and if the event is as likely to occur as not then the probability is 0.5 and the odds are 1:1. While we will most often use the probability scale, odds are sometimes more convenient because they are expressed on a multiplicative scale.

Background 2. The basics of probability.

Long before machine learning came into existence, philosophers knew that generalising from particular cases to general rules is not a well-posed problem with well-defined solutions. Such inference by generalisation is called *induction* and is to be contrasted with *deduction*, which is the kind of reasoning that applies to problems with well-defined correct solutions. There are many versions of this so-called *problem of induction*. One version is due to the eighteenth-century Scottish philosopher David Hume, who claimed that the only justification for induction is itself inductive: since it appears to work for certain inductive problems, it is expected to work for all inductive problems. This doesn't just say that induction cannot be deductively justified but that its justification is circular, which is much worse.

A related problem is stated by the *no free lunch theorem*, which states that no learning algorithm can outperform another when evaluated over all possible classification problems, and thus the performance of any learning algorithm, over the set of all possible learning problems, is no better than random guessing. Consider, for example, the 'guess the next number' questions popular in psychological tests: what comes after 1, 2, 4, 8, ...? If all number sequences are equally likely, then there is no hope that we can improve – on average – on random guessing (I personally always answer '42' to such questions). Of course, some sequences are very much more likely than others, at least in the world of psychological tests. Likewise, the distribution of learning problems in the real world is highly non-uniform. The way to escape the curse of the no free lunch theorem is to find out more about this distribution and exploit this knowledge in our choice of learning algorithm.

Background 1.1. Problems of induction and free lunches.

1.2 Models: the output of machine learning

Models form the central concept in machine learning as they are what is being learned from the data, in order to solve a given task. There is a considerable – not to say bewildering – range of machine learning models to choose from. One reason for this is the ubiquity of the tasks that machine learning aims to solve: classification, regression, clustering, association discovery, to name but a few. Examples of each of these tasks can be found in virtually every branch of science and engineering. Mathematicians, engineers, psychologists, computer scientists and many others have discovered – and sometimes rediscovered – ways to solve these tasks. They have all brought their

Transformations in d-dimensional Cartesian coordinate systems can be conveniently represented by means of matrix notation. Let $\mathbf{x}$ be a d-vector representing a data point, then $\mathbf{x}+\mathbf{t}$ is the resulting point after *translating* over $\mathbf{t}$ (another d-vector). Translating a set of points over $\mathbf{t}$ can be equivalently understood as translating the origin over $-\mathbf{t}$. Using *homogeneous coordinates* – the addition of an extra dimension set to 1 – translations can be expressed by matrix multiplication: e.g., in two dimensions we have

$$\mathbf{x}^\circ = \begin{pmatrix} 1 \\ x_1 \\ x_2 \end{pmatrix} \qquad \mathbf{T} = \begin{pmatrix} 1 & 0 & 0 \\ t_1 & 1 & 0 \\ t_2 & 0 & 1 \end{pmatrix} \qquad \mathbf{T}\mathbf{x}^\circ = \begin{pmatrix} 1 \\ x_1 + t_1 \\ x_2 + t_2 \end{pmatrix}$$

A *rotation* is defined by any d-by-d matrix $\mathbf{D}$ whose transpose is its inverse (which means it is orthogonal) and whose determinant is 1. In two dimensions a rotation matrix can be written as $\mathbf{R} = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}$, representing a clockwise rotation over angle θ about the origin. For instance, $\begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$ is a 90 degrees clockwise rotation.

A *scaling* is defined by a diagonal matrix; in two dimensions $\mathbf{S} = \begin{pmatrix} s_1 & 0 \\ 0 & s_2 \end{pmatrix}$. A *uniform scaling* applies the same scaling factor s in all dimensions and can be written as $s\mathbf{I}$, where $\mathbf{I}$ is the identity matrix. Notice that a uniform scaling with scaling factor -1 is a rotation (over 180 degrees in the two-dimensional case).

A common scenario which utilises all these transformations is the following. Given an n-by-d matrix $\mathbf{X}$ representing n data points in d-dimensional space, we first calculate the centre of mass or mean vector $\boldsymbol{\mu}$ by averaging each column. We then zero-centre the data set by subtracting $-\boldsymbol{\mu}$ from each row, which corresponds to a translation. Next, we rotate the data such that as much variance (a measure of the data's 'spread' in a certain direction) as possible is aligned with our coordinate axes; this can be achieved by a matrix transformation known as ☞*principal component analysis*, about which you will learn more in Chapter 10. Finally, we scale the data to unit variance along each coordinate.

Background 1.2. Linear transformations.

rather than being derived from a global model built from the entire data set.

There is a nice relationship between Euclidean distance and the mean of a set of

---

Random variables describe possible outcomes of a random process. They can be either discrete (e.g., the possible outcomes of rolling a die are $\{1,2,3,4,5,6\}$) or continuous (e.g., the possible outcomes of measuring somebody's weight in kilograms). Random variables do not need to range over integer or real numbers, but it does make the mathematics quite a bit simpler so that is what we assume here.

If X is a discrete random variable with probability distribution $P(X)$ then the *expected value* of X is $\mathbb{E}[X] = \sum_x xP(x)$. For instance, the expected value of tossing a fair die is $1\cdot\frac{1}{6} + 2\cdot\frac{1}{6} + \ldots + 6\cdot\frac{1}{6} = 3.5$. Notice that this is not actually a possible outcome. For a continuous random variable we need to replace the sum with an integral, and the probability distribution with a probability density function: $\mathbb{E}[X] = \int_{-\infty}^{+\infty} xp(x)\,dx$. The idea of this rather abstract concept is that if we take a sample $x_1,\ldots,x_n$ of outcomes of the random process, the expected value is what we expect the *sample mean* $\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ to be – this is the celebrated *law of large numbers* first proved by Jacob Bernoulli in 1713. For this reason the expected value is often called the *population mean*, but it is important to realise that the latter is a theoretical value, while the sample mean is an empirical *estimate* of that theoretical value.

The expectation operator can be applied to functions of random variables. For instance, the (population) *variance* of a discrete random variable is defined as $\mathbb{E}\left[(X-\mathbb{E}[X])^2\right] = \sum_x (x-\mathbb{E}[X])^2 P(x)$ – this measures the spread of the distribution around the expected value. Notice that

$$\mathbb{E}\left[(X-\mathbb{E}[X])^2\right] = \sum_x (x-\mathbb{E}[X])^2 P(x) = \mathbb{E}\left[X^2\right] - \mathbb{E}[X]^2$$

We can similarly define the *sample variance* as $\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i-\overline{x})^2$, which decomposes as $\frac{1}{n}\sum_{i=1}^{n} x_i^2 - \overline{x}^2$. You will sometimes see the sample variance defined as $\frac{1}{n-1}\sum_{i=1}^{n}(x_i-\overline{x})^2$: dividing by $n-1$ rather than n results in a slightly larger estimate, which compensates for the fact that we are calculating the spread around the sample mean rather than the population mean.

The (population) *covariance* between two discrete random variables X and Y is defined as $\mathbb{E}[(X-\mathbb{E}[X])(Y-\mathbb{E}[Y])] = \mathbb{E}[X\cdot Y] - \mathbb{E}[X]\cdot\mathbb{E}[Y]$ The variance of X is a special case of this, with $Y = X$. Unlike the variance, the covariance can be positive as well as negative. Positive covariance means that both variables tend to increase or decrease together; negative covariance means that if one variable increases, the other tends to decrease. If we have a sample of pairs of values of X and Y, *sample covariance* is defined as $\frac{1}{n}\sum_{i=1}^{n}(x_i-\overline{x})(y_i-\overline{y}) = \frac{1}{n}\sum_{i=1}^{n} x_i y_i - \overline{x}\,\overline{y}$. By dividing the covariance between X and Y by $\sqrt{\sigma_X^2\sigma_Y^2}$ we obtain the *correlation coefficient*, which is a number between -1 and $+1$.

Background 1.3. Expectations and estimators.

are indicative of ham, whereas e-mails with both terms are indicative of spam. Put differently, within the spam class these features are positively correlated, while within the ham class they are negatively correlated. In such a case, ignoring these interactions will be detrimental for classification performance. In other cases, feature correlations may obscure the true model – we shall see examples of this later in the book. On the other hand, feature correlation sometimes helps us to zoom in on the relevant part of the instance space.

There are other ways in which features can be related. Consider the following three features that can be true or false of a molecular compound:

1. it has a carbon in a six-membered aromatic ring;
2. it has a carbon with a partial charge of -0.13;
3. it has a carbon in a six-membered aromatic ring with a partial charge of -0.13.

We say that the third feature is more *specific* (or less *general*) than the other two, because if the third feature is true, then so are the first and the second. However, the converse does not hold: if both first and second feature are true, the third feature may still be false (because the carbon in the six-membered ring may not be the same as the one with a partial charge of -0.13). We can exploit these relationships when searching for features to add to our logical model. For instance, if we find that the third feature is true of a particular negative example that we're trying to exclude, then there is no point in considering the more general first and second features, because they will not help us in excluding the negative either. Similarly, if we find that the first feature is false of a particular positive we're trying to include, there is no point in considering the more specific third feature instead. In other words, these relationships help us to structure our search for predictive features.

1.4 Summary and outlook

My goal in this chapter has been to take you on a tour to admire the machine learning landscape, and to raise your interest sufficiently to want to read the rest of the book. Here is a summary of the things we have been looking at.

☞ Machine learning is about using the right features to build the right models that achieve the right tasks. These tasks include: binary and multi-class classification, regression, clustering and descriptive modelling. Models for the first few of these tasks are learned in a supervised fashion requiring labelled training data. For instance, if you want to train a spam filter using machine learning, you need a training set of e-mails labelled spam and ham. If you want to know how good the model is you also need labelled test data that is distinct from the training data, as evaluating your model on the data it was trained on will paint too rosy a picture: a test set is needed to expose any overfitting that occurs.

☞ Unsupervised learning, on the other hand, works with unlabelled data and so there is no test data as such. For instance, to evaluate a particular partition of data into clusters, one can calculate the average distance from the cluster centre. Other forms of unsupervised learning include learning associations (things that tend to occur together) and identifying hidden variables such as film genres. Overfitting is also a concern in unsupervised learning: for instance, assigning each data point its own cluster will reduce the average distance to the cluster centre to zero, yet is clearly not very useful.

☞ On the output side we can distinguish between predictive models whose outputs involve the target variable and descriptive models which identify interesting structure in the data. Often, predictive models are learned in a supervised setting while descriptive models are obtained by unsupervised learning methods, but there are also examples of supervised learning of descriptive models (e.g., subgroup discovery which aims at identifying regions with an unusual class distribution) and unsupervised learning of predictive models (e.g., predictive clustering where the identified clusters are interpreted as classes).

☞ We have loosely divided machine learning models into geometric models, probabilistic models and logical models. Geometric models are constructed in Cartesian instance spaces, using geometric concepts such as planes and distances. The prototypical geometric model is the basic linear classifier, which constructs a decision plane orthogonal to the line connecting the positive and negative centres of mass. Probabilistic models view learning as a process of reducing uncertainty using data. For instance, a Bayesian classifier models the posterior distribution $P(Y|X)$ (or its counterpart, the likelihood function $P(X|Y)$) which tells me the class distribution Y after observing the feature values X. Logical models are the most 'declarative' of the three, employing if–then rules built from logical conditions to single out homogeneous areas in instance space.

☞ We have also introduced a distinction between grouping and grading models. Grouping models divide the instance space into segments which are determined at training time, and hence have a finite resolution. On each segment, grouping models usually fit a very simple kind of model, such as 'always predict this class'. Grading models fit a more global model, graded by the location of an instance in instance space (typically, but not always, a Cartesian space). Logical models are typical examples of grouping models, while geometric models tend to be grading in nature, although this distinction isn't clear-cut. While this sounds very abstract at the moment, the distinction will become much clearer when we discuss coverage curves in the next chapter.

☞ Last but not least, we have discussed the role of features in machine learning. No model can exist without features, and sometimes a single feature is enough to build a model. Data doesn't always come with ready-made features, and often we have to transform or even construct features. Because of this, machine learning is often an iterative process: we only know we have captured the right features after we have constructed the model, and if the model doesn't perform satisfactorily we need to analyse its performance to understand in what way the features need to be improved.

What you'll find in the rest of the book

In the next nine chapters, we will follow the structure laid out above, and look in detail at

☞ machine learning tasks in Chapters 2 and 3;

☞ logical models: concept learning in Chapter 4, tree models in Chapter 5 and rule models in Chapter 6;

☞ geometric models: linear models in Chapter 7 and distance-based models in Chapter 8;

☞ probabilistic models in Chapter 9; and

☞ features in Chapter 10.

Chapter 11 is devoted to techniques for training 'ensembles' of models that have certain advantages over single models. In Chapter 12 we will consider a number of methods for what machine learners call 'experiments', which involve training and evaluating models on real data. Finally, in the Epilogue we will wrap up the book and take a look ahead.

We briefly review some important concepts from discrete mathematics. A *set* is a collection of objects, usually of the same kind (e.g., the set of all natural numbers $\mathbb{N}$ or the set of real numbers $\mathbb{R}$). We write $x \in A$ if x is an element of set A, and $A \subseteq B$ if all elements of A are also elements of B (this includes the possibility that A and B are the same set, which is equivalent to $A \subseteq B$ and $B \subseteq A$). The *intersection* and *union* of two sets are defined as $A \cap B = \{x|x \in A \text{ and } x \in B\}$ and $A \cup B = \{x|x \in A \text{ or } x \in B\}$. The *difference* of two sets is defined as $A \setminus B = \{x|x \in A \text{ and } x \notin B\}$. It is customary to fix a *universe of discourse* U such that all sets under consideration are subsets of U. The *complement* of a set A is defined as $\overline{A} = U \setminus A$. Two sets are *disjoint* if their intersection is empty: $A \cap B = \emptyset$. The *cardinality* of a set A is its number of elements and is denoted $|A|$. The *powerset* of a set A is the set of all its subsets $2^A = \{B|B \subseteq A\}$; its cardinality is $|2^A| = 2^{|A|}$. The *characteristic function* of a set A is the function $f : U \rightarrow \{\mathsf{true}, \mathsf{false}\}$ such that $f(x) = \mathsf{true}$ if $x \in A$ and $f(x) = \mathsf{false}$ if $x \in U \setminus A$.

If A and B are sets, the *Cartesian product* $A \times B$ is the set of all pairs $\{(x, y)|x \in A \text{ and } y \in B\}$; this generalises to products of more than two sets. A (binary) *relation* is a set of pairs $R \subseteq A \times B$ for some sets A and B; if $A = B$ we say the relation is over A. Instead of $(x, y) \in R$ we also write xRy. A relation over A is (*i*) *reflexive* if xRx for all $x \in A$; (*ii*) *symmetric* if xRy implies yRx for all $x, y \in A$; (*iii*) *antisymmetric* if xRy and yRx implies $x = y$ for all $x, y \in A$; (*iv*) *transitive* if xRy and yRz implies xRz for all $x, y, z \in A$. (*v*) *total* if xRy or yRx for all $x, y \in A$.

A *partial order* is a binary relation that is reflexive, antisymmetric and transitive. For instance, the *subset* relation $\subseteq$ is a partial order. A *total order* is a binary relation that is total (hence reflexive), antisymmetric and transitive. The $\leq$ relation on real numbers is a total order. If xRy or yRx we say that x and y are *comparable*; otherwise they are *incomparable*. An *equivalence relation* is a binary relation $\equiv$ that is reflexive, symmetric and transitive. The *equivalence class* of x is $[x] = \{y|x \equiv y\}$. For example, the binary relation 'contains the same number of elements as' over any set is an equivalence relation. Any two equivalence classes are disjoint, and the union of all equivalence classes is the whole set – in other words, the set of all equivalence classes forms a *partition* of the set. If $A_1, \ldots, A_n$ is a partition of a set A, i.e. $A_1 \cup \ldots \cup A_n = A$ and $A_i \cap A_j = \emptyset$ for all $i \neq j$, we write $A = A_1 \uplus \ldots \uplus A_n$.

To illustrate this, let T be a feature tree, and define a relation $\sim_T \subseteq \mathscr{X} \times \mathscr{X}$ such that $x \sim_T x'$ if and only if x and x' are assigned to the same leaf of feature tree T, then $\sim_T$ is an equivalence relation, and its equivalence classes are precisely the instance space segments associated with T.

Background 2.1. Useful concepts from discrete mathematics.

The sections in this chapter are devoted to the first three scenarios in Table 2.1:

2.4 Binary classification and related tasks: Summary and further reading

In this chapter we have looked at binary classification, a ubiquitous task that forms the starting point of a lot of work in machine learning. Although we haven't talked much about learning in this chapter, my philosophy is that you will reach a better understanding of machine learning models and algorithms if you first study the tasks that these models are meant to address.

☞ In Section 2.1 we defined the binary classification task and introduced an important tool to assess performance at such a task, namely the two-by-two contingency table. A wide range of performance indicators are derived from the counts in a contingency table. I introduced the coverage plot, which visualises a contingency table as a rectangle with size *Pos* up and size *Neg* across, and within that rectangle a point with y-coordinate *TP* and x-coordinate *FP*. We can visualise several models evaluated on the same data set by several points, and use the fact that accuracy is constant along line segments with slope 1 to visually rank these classifiers on accuracy. Alternatively, we can normalise the rectangle to be a unit square with true and false positive rate on the axes. In this so-called ROC space, line segments with slope 1 (i.e., those parallel to the ascending diagonal) connect points with the same average recall (sometimes also called macro-accuracy). The use of these kinds of plot in machine learning was pioneered by Provost and Fawcett (2001). Unnormalised coverage plots were introduced by Fürnkranz and Flach (2003).

☞ Section 2.2 considered the more general task of calculating a score for each example (or a vector of scores in the general case of more than two classes). While the scale on which scores are expressed is unspecified, it is customary to put the decision threshold at $\hat{s}(x) = 0$ and let the sign of the score stand for the prediction (positive or negative). Multiplying the score with the true class gives us the margin, which is positive for a correct prediction and negative for an incorrect one. A loss function determines how much negative margins are penalised and positive margins rewarded. The advantage of working with convex and continuously differentiable 'surrogate' loss functions (rather than with 0–1 loss, which is the loss function we ultimately want to optimise) is that this often leads to more tractable optimisation problems.

☞ Alternatively, we can ignore the scale on which scores are measured altogether and only work with their order. Such a ranker is visualised in coverage or ROC space by a piecewise continuous curve. For grouping models the line segments in these curves correspond to instance space segments (e.g., the leaves of a tree model) whereas for grading models there is a segment for each unique score assigned by the model. The area under the ROC curve gives the ranking accuracy (an estimate of the probability that a random positive is ranked before a random negative) and is known in statistics as the Wilcoxon-Mann-Whitney statistic These curves can be used to find a suitable operating point by translating the operating condition (class and cost distribution) into an isometric in ROC or coverage space. The origins of ROC curves are in signal detection theory (Egan, 1975); accessible introductions can be found in (Fawcett, 2006; Flach, 2010*b*).

☞ In Section 2.3 we looked at scoring models whose scores can be interpreted as estimates of the probability that the instance belongs to a particular class. Such models were pioneered in forecasting theory by Brier (1950) and Murphy and Winkler (1984), among others. We can assess the quality of class probability estimates by comparing them to the 'ideal' probabilities (1 for a positive, 0 for a negative) and taking mean squared error. Since there is no reason why the true probabilities should be categorical this is quite a crude assessment, and decomposing it into calibration loss and refinement loss provides useful additional information. We have also seen a very useful trick for smoothing relative frequency estimates of probabilities by adding pseudo-counts, either uniformly distributed (Laplace correction) or according to a chosen prior (m-estimate). Finally, we have seen how we can use the ROC convex hull to obtain calibrated class probability estimates. The approach has its roots in isotonic regression (Best and Chakravarti, 1990) and was introduced to the machine learning community by Zadrozny and Elkan (2002). Fawcett and Niculescu-Mizil (2007) and Flach and Matsubara (2007) show that the approach is equivalent to calibration by means of the ROC convex hull. (Note that in this chapter we have seen two different uses of the term 'convex': one in relation to loss functions, where convexity means that linear interpolation between any two points on the curve depicting the loss function will never result in a point below the curve; and the other in relation to the ROC convex hull, where it refers to the linearly interpolated boundary of a convex set which envelopes all points in the set.)

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.

We clearly see a relationship with subgroup discovery in that association rules also identify subsets that have a different distribution when compared with the full data set, namely with respect to the then-part of the rule. The difference is that the then-part is not a fixed target variable but it is found as part of the discovery process. Both subgroup discovery and association rule discovery will be discussed in the context of rule learning in Section 6.3.

3.4 Beyond binary classification: Summary and further reading

While binary classification is an important task in machine learning, there are many other relevant tasks and in this chapter we looked at a number of them.

☞ In Section 3.1 we considered classification tasks with more than two classes. We shall see in the coming chapters that some models handle this situation very naturally, but if our models are essentially two-class (such as linear models) we have to approach it via a combination of binary classification tasks. One key idea is the use of a code matrix to combine the results of several binary classifiers, as proposed by Dietterich and Bakiri (1995) under the name 'error-correcting output codes' and developed by Allwein *et al.* (2000). We also looked at ways to obtain scores for more than two classes and to evaluate those scores using multi-class adaptations of the area under the ROC curve. One of these multi-class extensions of AUC was proposed and analysed by Hand and Till (2001). The heuristic procedure for reweighting multi-class scores in Example 3.6 on p.89 was proposed by Lachiche and Flach (2003); Bourke *et al.* (2008) demonstrated that it achieves good performance in comparison with a number of alternative approaches.

☞ Section 3.2 was devoted to regression: predicting a real-valued target value. This is a classical data analysis problem that was already studied by Carl Friedrich Gauss in the late eighteenth century. It is natural to use a quadratic loss function on the residuals, although this carries with it a certain sensitivity to outliers. Grading models are most common here, although it is also possible to learn a grouping model that divides the instance space into segments that admit a simple local model. Since it is often possible to fit a set of points exactly (e.g., with a high-degree polynomial), care must be taken to avoid overfitting. Finding the right balance between over- and underfitting is sometimes called the bias–variance dilemma; an extensive discussion (including the dartboard metaphor) can be found in Rajnarayan and Wolpert (2010).

☞ In Section 3.3 we considered unsupervised and descriptive learning tasks. We saw that in descriptive learning the task and learning problem coincide. A clustering model can be either predictive or descriptive: in the former case it is meant to construct classes in a wholly unsupervised manner, after which the learned model can be applied to unseen data in the usual way. Descriptive clustering, on the other hand, only applies to the data at hand. It should be noted that the distinction between predictive and descriptive clustering is not universally recognised in the literature; sometimes the term 'predictive clustering' is used in the slightly different sense of clustering simultaneously on the target variable and the features (Blockeel *et al.*, 1998).

☞ Like descriptive clustering, association rule discovery is another descriptive task which is wholly unsupervised. It was introduced by Agrawal, Imielinski and Swami (1993) and has given rise to a very large body of work in the data mining literature. Subgroup discovery is a form of supervised learning of descriptive models aimed at finding subsets of the data with a significantly different distribution of the target variable. It was first studied by Klösgen (1996) and extended to the more general notion of exceptional model mining in order to deal with, e.g., real-valued target variables by Leman *et al.* (2008). More generally, unsupervised learning of descriptive models is a large subject that was pioneered by Tukey (1977).

The simplest logical expressions are equalities of the form Feature = Value and, for numerical features, inequalities of the form Feature < Value; these are called *literals*. Complex Boolean expressions can be built using logical connectives: *conjunction* $\land$, *disjunction* $\lor$, *negation* $\neg$ and *implication* $\rightarrow$. The following equivalences hold (the left two are called the *De Morgan laws*):

$$\neg(A \land B) \equiv \neg A \lor \neg B \qquad \neg\neg A \equiv A$$

$$\neg(A \lor B) \equiv \neg A \land \neg B \qquad A \rightarrow B \equiv \neg A \lor B$$

If Boolean expression A is true of instance x, we say that A *covers* x. The set of instances covered by expression A is called its *extension* and denoted $\mathscr{X}_A = \{x \in \mathscr{X} | A \text{ covers } x\}$, where $\mathscr{X}$ denotes the instance space which acts as the universe of discourse (see Background 2.1 on p.51). There is a direct correspondence between logical connectives and operations on sets: e.g., $\mathscr{X}_{A \land B} = \mathscr{X}_A \cap \mathscr{X}_B$, $\mathscr{X}_{A \lor B} = \mathscr{X}_A \cup \mathscr{X}_B$ and $\mathscr{X}_{\neg A} = \mathscr{X} \setminus \mathscr{X}_A$. If $\mathscr{X}_A \supseteq \mathscr{X}_{A'}$, we say that A is *at least as general as* A', and if in addition $\mathscr{X}_A \not\subseteq \mathscr{X}_{A'}$ we say that A is *more general than* A'. This *generality ordering* is a partial order on logical expressions as defined in Background 2.1. (More precisely: it is a partial order on the equivalence classes of the relation of logical equivalence $\equiv$.)

A *clause* is an implication $P \rightarrow Q$ such that P is a conjunction of literals and Q is a disjunction of literals. Using the equivalences above we can rewrite such an implication as

$$(A \land B) \rightarrow (C \lor D) \equiv \neg(A \land B) \lor (C \lor D) \equiv \neg A \lor \neg B \lor C \lor D$$

and hence a clause can equivalently be seen as a disjunction of literals or their negations. Any logical expression can be rewritten as a conjunction of clauses; this is referred to as *conjunctive normal form* (*CNF*). Alternatively, any logical expression can be written as a disjunction of conjunctions of literals or their negation; this is called *disjunctive normal form* (*DNF*). A *rule* is a clause $A \rightarrow B$ where B is a single literal; this is also often referred to as a *Horn clause*, after the American logician Alfred Horn.

Background 4.1. Some logical concepts and notation.

we consider tree and rule models, which go considerably beyond concept learning as they can handle multiple classes, probability estimation, regression, as well as clustering tasks.

4.5 Concept learning: Summary and further reading

In this chapter we looked at methods for inductive concept learning: the process of constructing a logical expression defining a set of objects from examples. This problem was a focus of early work in artificial intelligence (Winston, 1970; Vere, 1975; Banerji, 1980), following the seminal work by psychologists Bruner, Goodnow and Austin (1956) and Hunt, Marin and Stone (1966).

☞ In Section 4.1 we considered the structure of the hypothesis space: the set of possible concepts. Every hypothesis has an extension (the set of instances it covers), and thus relationships between extensions such as subset relationships carry over to the hypothesis space. This gives the hypothesis space a lattice structure: a partial order with least upper bounds and greatest lower bounds. In particular, the LGG is the least upper bound of a set of instances, and is the most conservative generalisation that we can learn from the data. The concept was defined in the context of first-order logic by Plotkin (1971), who showed that it was the mathematical dual of the deductive operation of unification. We can extend the hypothesis language with internal disjunction among values of a feature, which creates a larger hypothesis space that still has a lattice structure. Internal disjunction is a common staple of attribute-value languages for learning following the work of Michalski (1973). For further pointers regarding hypothesis language and hypothesis space the reader is referred to (Blockeel, 2010*a*,*b*).

☞ Section 4.2 defined complete and consistent hypotheses as concepts that cover all positive examples and no negative examples. The set of complete and consistent concepts is called the version space, a notion introduced by Mitchell (1977). The version space can be summarised by its least general and most general members, since any concept between one least general hypothesis and another most general one is also complete and consistent. Alternatively, we can describe the version space by all paths from a least general to a most general hypothesis. Such upward paths give rise to a coverage curve which describes the extension of each concept on the path in terms of covered positives and negatives. Concept learning can then be seen as finding an upward path that goes through ROC heaven. Syntactically different concepts can have the same extension in a particular data set: a closed concept is the most specific one of these (technically, the LGG of the instances in its extension). The notion is studied in formal concept analysis (Ganter and Wille, 1999) and was introduced in a data mining context by Pasquier, Bastide, Taouil and Lakhal (1999); Garriga, Kralj and Lavrač (2008) investigate its usefulness for labelled data.

☞ In Section 4.3 we discussed the Horn algorithm for learning concepts described by conjunctions of Horn rules, first published in Angluin *et al.* (1992). The algorithm makes use of a membership oracle, which can be seen as an early form of active learning (Cohn, 2010; Dasgupta, 2010). Horn theories are superficially similar to classification rule models which will be studied in Chapter 6. However, there is an important difference, since those classification rules have the target variable in the then-part of the rule, while the Horn clauses we are looking at here can have any literal in the then-part. In fact, in this chapter the target variable is not part of the logical language at all. This setting is sometimes called *learning from interpretations*, since examples are truth-value assignments to our theory. The classification rule setting is called *learning from entailment*, since in order to find out whether a particular rule covers an example we need to apply logical inference. De Raedt (1997) explains and explores the differences between these two settings. Further introductions to first-order logic and its use in learning are given by Flach (2010*a*) and De Raedt (2010).

☞ Section 4.4 briefly reviewed some basic concepts and results in learnability theory. My account partly followed Mitchell (1997, Chapter 7); another excellent introduction is given by Zeugmann (2010). PAC-learnability, which allows an error rate of ϵ and a failure rate of δ, was introduced in a seminal paper by Valiant (1984). Haussler (1988) derived the sample complexity for complete and consistent learners (Equation 4.1), which is linear in $1/\epsilon$ and logarithmic in $1/\delta$ and the size of the hypothesis space. The VC-dimension as a measure of the capacity of a hypothesis language was introduced by Vapnik and Chervonenkis (1971) in order to quantify the difference between training error and true error. This allows a statement of the sample complexity in terms of the VC-dimension (Equation 4.2) which is due to Blumer, Ehrenfeucht, Haussler and Warmuth (1989). This same paper proved that a model class is PAC-learnable if and only if its VC-dimension is finite.

The *variance* of a set of numbers $X \subseteq \mathbb{R}$ is defined as the average squared difference from the mean:

$$\mathrm{Var}(X) = \frac{1}{|X|}\sum_{x\in X}(x-\overline{x})^2$$

where $\overline{x} = \frac{1}{|X|}\sum_{x\in X} x$ is the mean of X. Expanding $(x-\overline{x})^2 = x^2 - 2\overline{x}x + \overline{x}^2$ this can be written as

$$\mathrm{Var}(X) = \frac{1}{|X|}\left(\sum_{x\in X} x^2 - 2\overline{x}\sum_{x\in X} x + \sum_{x\in X}\overline{x}^2\right) = \frac{1}{|X|}\left(\sum_{x\in X} x^2 - 2\overline{x}|X|\overline{x} + |X|\overline{x}^2\right) = \frac{1}{|X|}\sum_{x\in X} x^2 - \overline{x}^2 \tag{5.3}$$

So the variance is the difference between the mean of the squares and the square of the mean.

It is sometimes useful to consider the average squared difference from another value $x' \in \mathbb{R}$, which can similarly be expanded:

$$\frac{1}{|X|}\sum_{x\in X}(x-x')^2 = \frac{1}{|X|}\left(\sum_{x\in X} x^2 - 2x'|X|\overline{x} + |X|x'^2\right) = \mathrm{Var}(X) + (x'-\overline{x})^2$$

The last step follows because from Equation 5.3 we have $\frac{1}{|X|}\sum_{x\in X} x^2 = \mathrm{Var}(X) + \overline{x}^2$.

Another useful property is that the average squared difference between any two elements of X is twice the variance:

$$\frac{1}{|X|^2}\sum_{x'\in X}\sum_{x\in X}(x-x')^2 = \frac{1}{|X|}\sum_{x'\in X}(\mathrm{Var}(X) + (x'-\overline{x})^2) = \mathrm{Var}(X) + \frac{1}{|X|}\sum_{x'\in X}(x'-\overline{x})^2 = 2\mathrm{Var}(X)$$

If $X \subseteq \mathbb{R}^d$ is a set of d-vectors of numbers, we can define the variance $\mathrm{Var}_i(X)$ for each of the d coordinates. We can then interpret the sum of variances $\sum_{i=1}^{d}\mathrm{Var}_i(X)$ as the average squared Euclidean distance of the vectors in X to their vector mean $\overline{\mathbf{x}} = \frac{1}{|X|}\sum_{\mathbf{x}\in X}\mathbf{x}$.

(You will sometimes see sample variance defined as $\frac{1}{|X|-1}\sum_{x\in X}(x-\overline{x})^2$, which is a somewhat larger value. This version arises if we are estimating the variance of a population from which X is a random sample: normalising by $|X|$ would underestimate the population variance because of differences between the sample mean and the population mean. Here, we are only concerned with assessing the spread of the given values X and not with some unknown population, and so we can ignore this issue.)

Background 5.1. Variations on variance.

So, in order to obtain a regression tree learning algorithm, we replace the impurity measure Imp in Algorithm 5.2 with the function Var. Notice that $\frac{1}{|Y|}\sum_{y\in Y} y^2$ is constant for a given set Y, and so minimising variance over all possible splits of a given parent is the same as maximising the weighted average of squared means in the chil-

In this example we used categorical features for splitting and numerical features for distance calculations. Indeed, in all tree examples considered so far we have only used categorical features for splitting.[7] In practice, numerical features are frequently used for splitting: all we need to do is find a suitable threshold t so that feature F can be turned into a binary split with conditions $F \geq t$ and $F < t$. Finding the optimal split point is closely related to ☞*discretisation* of numerical features, a topic we will look at in detail in Chapter 10. For the moment, the following observations give some idea how we can learn a threshold on a numerical feature:

- ☞ Although in theory there are infinitely many possible thresholds, in practice we only need to consider values separating two examples that end up next to each other if we sort the training examples on increasing (or decreasing) value of the feature.
- ☞ We only consider consecutive examples of different class if our task is classification, whose target values are sufficiently different if our task is regression, or whose dissimilarity is sufficiently large if our task is clustering.
- ☞ Each potential threshold can be evaluated as if it were a distinct binary feature.

5.4 Tree models: Summary and further reading

Tree-based data structures are ubiquitous in computer science, and the situation is no different in machine learning. Tree models are concise, easy to interpret and learn, and can be applied to a wide range of tasks, including classification, ranking, probability estimation, regression and clustering. The tree-based classifier for human pose recognition in the Microsoft Kinect motion sensing device is described in Shotton *et al.* (2011).

- ☞ I introduced the feature tree as the common core for all these tree-based models, and the recursive GrowTree algorithm as a generic divide-and-conquer algorithm that can be adapted to each of these tasks by suitable choices for the functions that test whether a data set is sufficiently homogeneous, find a suitable label if it is, and find the best feature to split on if it isn't.

- ☞ Using a feature tree to predict class labels turns them into decision trees, the subject of Section 5.1. There are two classical accounts of decision trees in machine learning, which are very similar algorithmically but differ in details such as heuristics and pruning strategies. Quinlan's approach was to use entropy as impurity measure, and progressed from the ID3 algorithm (Quinlan, 1986), which

[7]Categorical features are features with a relatively small set of discrete values. Technically, they distinguish themselves from numerical features by not having a scale or an ordering. This is further explored in Chapter 10.

itself was inspired by Hunt, Marin and Stone (1966), to the sophisticated C4.5 system (Quinlan, 1993). The CART approach stands for 'classification and regression trees' and was developed by Breiman, Friedman, Olshen and Stone (1984); it uses the Gini index as impurity measure. The $\sqrt{\text{Gini}}$ impurity measure was introduced by Dietterich, Kearns and Mansour (1996), and is hence sometimes referred to as *DKM*. The geometric construction to find $\text{Imp}(\{D_1, D_2\})$ in Figure 5.2 (right) was also inspired by that paper.

☞ Employing the empirical distributions in the leaves of a feature tree in order to build rankers and probability estimators as described in Section 5.2 is a much more recent development (Ferri *et al.*, 2002; Provost and Domingos, 2003). Experimental results demonstrating that better probability estimates are obtained by disabling tree pruning and smoothing the empirical probabilities by means of the Laplace correction are presented in the latter paper and corroborated by Ferri *et al.* (2003). The extent to which decision tree splitting criteria are insensitive to unbalanced classes or misclassification costs was studied and explained by Drummond and Holte (2000) and Flach (2003). Of the three splitting criteria mentioned above, only $\sqrt{\text{Gini}}$ is insensitive to such class and cost imbalance.

☞ Tree models are grouping models that aim to minimise diversity in their leaves, where the appropriate notion of diversity depends on the task. Very often diversity can be interpreted as some kind of variance, an idea that already appeared in (Breiman *et al.*, 1984) and was revisited by Langley (1994), Kramer (1996) and Blockeel, De Raedt and Ramon (1998), among others. In Section 5.3 we saw how this idea can be used to learn regression and clustering trees (glossing over many important details, such as when we should stop splitting nodes).

It should be kept in mind that the increased expressivity of tree models compared with, say, conjunctive concepts means that we should safeguard ourselves against overfitting. Furthermore, the greedy divide-and-conquer algorithm has the disadvantage that small changes in the training data may lead to a different choice of the feature at the root of the tree, which will influence the choice of feature at subsequent splits. We will see in Chapter 11 how methods such as bagging can be applied to help reduce this kind of model variance.

of hypotheses and target predicates that are to be learned, and introduces computational challenges such as non-termination. However, this doesn't mean that it cannot be done. Related techniques can be used to learn multiple, interrelated predicates at once, and to invent new background predicates that are completely unobserved.

6.5 Rule models: Summary and further reading

In a decision tree, a branch from root to a leaf can be interpreted as a conjunctive classification rule. Rule models generalise this by being more flexible about the way in which several rules are combined into a model. The typical rule learning algorithm is the covering algorithm, which iteratively learns one rule and then removes the examples covered by that rule. This approach was pioneered by Michalski (1975) with his AQ system, which became highly developed over three decades (Wojtusiak *et al.*, 2006). General overviews are provided by Fürnkranz (1999, 2010) and Fürnkranz, Gamberger and Lavrač (2012). Coverage plots were first used by Fürnkranz and Flach (2005) to achieve a better understanding of rule learning algorithms and demonstrate the close relationship (and in many cases, equivalence) of commonly used search heuristics.

☞ Rules can overlap and thus we need a strategy to resolve potential conflicts between rules. One such strategy is to combine the rules in an ordered rule list, which was the subject of Section 6.1. Rivest (1987) compares this approach with decision trees, calling the rule-based model a decision list (I prefer the term 'rule list' as it doesn't carry a suggestion that the elements of the list are single literals). Well-known rule list learners include CN2 (Clark and Niblett, 1989) and Ripper (Cohen, 1995), the latter being particularly effective at avoiding overfitting through incremental reduced-error pruning (Fürnkranz and Widmer, 1994). Also notable is the Opus system (Webb, 1995), which distinguishes itself by performing a complete search through the space of all possible rules.

☞ In Section 6.2 we looked at unordered rule sets as an alternative to ordered rule lists. The covering algorithm is adapted to learn rules for a single class at a time, and to remove only covered examples of the class currently under consideration. CN2 can be run in unordered mode to learn rule sets (Clark and Boswell, 1991). Conceptually, both rule lists and rule sets are special cases of rule trees, which distinguish all possible Boolean combinations of a given set of rules. This allows us to see that rule lists lead to fewer instance space segments than rule sets (over the set of rules); on the other hand, rule list coverage curves can be made convex on the training set, whereas rule sets need to estimate the class distribution in the regions where rules overlap.

☞ Rule models can be used for descriptive tasks, and in Section 6.3 we considered rule learning for subgroup discovery. The weighted covering algorithm was introduced as an adaption of CN2 by Lavrač, Kavšek, Flach and Todorovski (2004); Abudawood and Flach (2009) generalise this to more than two classes. Algorithm 6.7 learns association rules and is adapted from the well-known Apriori algorithm due to Agrawal, Mannila, Srikant, Toivonen and Verkamo (1996). There is a very wide choice of alternative algorithms, surveyed by Han *et al.* (2007). Association rules can also be used to build effective classifiers (Liu *et al.*, 1998; Li *et al.*, 2001).

☞ The topic of first-order rule learning briefly considered in Section 6.4 has been studied for the last 40 years and has a very rich history. De Raedt (2008) provides an excellent recent introduction, and an overview of recent advances and open problems is provided by Muggleton *et al.* (2012). Flach (1994) gives an introduction to Prolog and also provides high-level implementations of some of the key techniques in inductive logic programming. The FOIL system by Quinlan (1990) implements a top–down learning algorithm similar to the one discussed here. The bottom–up technique was pioneered in the Golem system (Muggleton and Feng, 1990) and further refined in Progol (Muggleton, 1995) and in Aleph (Srinivasan, 2007), two of the most widely used ILP systems. First-order rules can also be learned in an unsupervised fashion, for example by Tertius which learns first-order clauses (not necessarily Horn) (Flach and Lachiche, 2001) and Warmr which learns first-order association rules (King *et al.*, 2001). Higher-order logic provides more powerful data types that can be highly beneficial in learning (Lloyd, 2003). A more recent development is the combination of probabilistic modelling with first-order logic, leading to the area of statistical relational learning (De Raedt and Kersting, 2010).

If x_1 and x_2 are two scalars or vectors of the same dimension and α and β are arbitrary scalars, then $\alpha x_1 + \beta x_2$ is called a *linear combination* of x_1 and x_2. If f is a *linear function* of x, then

$$f(\alpha x_1 + \beta x_2) = \alpha f(x_1) + \beta f(x_2)$$

In words, the function value of a linear combination of some inputs is a linear combination of their function values. As a special case, if $\beta = 1 - \alpha$ we are taking a weighted average of x_1 and x_2, and the linearity of f then means that the function value of the weighted average is the weighted average of the function values.

Linear functions take particular forms, depending on the domain and codomain of f. If x and $f(x)$ are scalars, it follows that f is of the form $f(x) = a + bx$ for some constants a and b; a is called the *intercept* and b the *slope*. If $\mathbf{x} = (x_1, \dots, x_d)$ is a vector and $f(\mathbf{x})$ is a scalar, then f is of the form

$$f(\mathbf{x}) = a + b_1 x_1 + \ldots + b_d x_d = a + \mathbf{b} \cdot \mathbf{x} \tag{7.1}$$

with $\mathbf{b} = (b_1, \dots, b_d)$. The equation $f(\mathbf{x}) = 0$ defines a plane in $\mathbb{R}^d$ perpendicular to the *normal vector* $\mathbf{b}$.

The most general case is where $f(\mathbf{x})$ is a d'-dimensional vector, in which case f is of the form $f(\mathbf{x}) = \mathbf{M}\mathbf{x} + \mathbf{t}$, where $\mathbf{M}$ is a d'-by-d matrix representing a *linear transformation* such as a rotation or a scaling, and $\mathbf{t}$ is a d'-vector representing a translation. In this case f is called an *affine transformation* (the difference between linear and affine transformations is that the former maps the origin to itself; notice that a linear function of the form of Equation 7.1 is a linear transformation only if the intercept is 0).

In all these forms we can avoid representing the intercept a or the translation $\mathbf{t}$ separately by using homogeneous coordinates. For instance, by writing $\mathbf{b}^\circ = (a, b_1, \dots, b_d)$ and $\mathbf{x}^\circ = (1, x_1, \dots, x_d)$ in Equation 7.1 we have $f(\mathbf{x}) = \mathbf{b}^\circ \cdot \mathbf{x}^\circ$ (see also Background 1.2 on p.24).

Examples of non-linear functions are the polynomials in x of degree $p > 1$: $g(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_p x^p = \sum_{i=0}^{p} a_i x^i$. Other non-linear functions can be approximated by a polynomial through their Taylor expansion. The *linear approximation* of a function g at x_0 is $g(x_0) + g'(x_0)(x - x_0)$, where $g'(x)$ is the derivative of x. A *piecewise linear* approximation is obtained by combining several linear approximations at different points x_0.

Background 7.1. Linear models.

simple as possible, but not simpler' that we introduced on p.30). Here are a couple of manifestations of this simplicity.

☞ Linear models are *parametric*, meaning that they have a fixed form with a small number of numeric parameters that need to be learned from data. This is

$\mathbf{X}$ usually denotes an n-by-d data matrix containing n instances in rows described by d features or variables in columns. $\mathbf{X}_{r\cdot}$ denotes the r-th row of $\mathbf{X}$, $\mathbf{X}_{\cdot c}$ denotes the c-th column, and $\mathbf{X}_{rc}$ denotes the entry in the r-th row and c-th column. We also use i and j to range over rows and columns, respectively. The j-th column mean is defined as $\mu_j = \frac{1}{n}\sum_{i=1}^{n}\mathbf{X}_{ij}$; $\boldsymbol{\mu}^{\mathrm{T}}$ is a row vector containing all column means. If $\mathbf{1}$ is an n-vector containing only ones, then $\mathbf{1}\boldsymbol{\mu}^{\mathrm{T}}$ is an n-by-d matrix whose rows are $\boldsymbol{\mu}^{\mathrm{T}}$; hence $\mathbf{X}' = \mathbf{X} - \mathbf{1}\boldsymbol{\mu}^{\mathrm{T}}$ has mean zero in each column and is referred to as the *zero-centred* data matrix.

The *scatter matrix* is the d-by-d matrix $\mathbf{S} = \mathbf{X}'^{\mathrm{T}}\mathbf{X}' = \left(\mathbf{X} - \mathbf{1}\boldsymbol{\mu}^{\mathrm{T}}\right)^{\mathrm{T}}\left(\mathbf{X} - \mathbf{1}\boldsymbol{\mu}^{\mathrm{T}}\right) = \mathbf{X}^{\mathrm{T}}\mathbf{X} - n\mathbf{M}$, where $\mathbf{M} = \boldsymbol{\mu}\boldsymbol{\mu}^{\mathrm{T}}$ is a d-by-d matrix whose entries are products of column means $\mathbf{M}_{jc} = \mu_j\mu_c$. The *covariance matrix* of $\mathbf{X}$ is $\boldsymbol{\Sigma} = \frac{1}{n}\mathbf{S}$ whose entries are the pairwise covariances $\sigma_{jc} = \frac{1}{n}\sum_{i=1}^{n}\left(\mathbf{X}_{ij} - \mu_j\right)\left(\mathbf{X}_{ic} - \mu_c\right) = \frac{1}{n}\left(\sum_{i=1}^{n}\mathbf{X}_{ij}\mathbf{X}_{ic} - \mu_i\mu_c\right)$. Two uncorrelated features have a covariance close to 0; positively correlated features have a positive covariance, indicating a certain tendency to increase or decrease together; a negative covariance indicates that if one feature increases, the other tends to decrease and vice versa. $\sigma_{jj} = \frac{1}{n}\sum_{i=1}^{n}\left(\mathbf{X}_{ij} - \mu_j\right)^2 = \frac{1}{n}\left(\sum_{i=1}^{n}\mathbf{X}_{ij}^2 - \mu_j^2\right)$ is the *variance* of column j, also denoted as σ_j^2. The variance is always positive and indicates the spread of the values of a feature around their mean.

A small example clarifies these definitions:

$$\mathbf{X} = \begin{pmatrix} 5 & 0 \\ 3 & 5 \\ 1 & 7 \end{pmatrix} \quad \mathbf{1}\boldsymbol{\mu}^{\mathrm{T}} = \begin{pmatrix} 3 & 4 \\ 3 & 4 \\ 3 & 4 \end{pmatrix} \quad \mathbf{X}' = \begin{pmatrix} 2 & -4 \\ 0 & 1 \\ -2 & 3 \end{pmatrix} \quad \mathbf{G} = \begin{pmatrix} 25 & 15 & 5 \\ 15 & 34 & 38 \\ 5 & 38 & 50 \end{pmatrix}$$

$$\mathbf{X}^{\mathrm{T}}\mathbf{X} = \begin{pmatrix} 35 & 22 \\ 22 & 74 \end{pmatrix} \quad \mathbf{M} = \begin{pmatrix} 9 & 12 \\ 12 & 16 \end{pmatrix} \quad \mathbf{S} = \begin{pmatrix} 8 & -14 \\ -14 & 26 \end{pmatrix} \quad \boldsymbol{\Sigma} = \begin{pmatrix} 8/3 & -14/3 \\ -14/3 & 26/3 \end{pmatrix}$$

We see that the two features are negatively correlated and that the second feature has the larger variance. Another way to calculate the scatter matrix is as a sum of outer products, one for each data point: $\mathbf{S} = \sum_{i=1}^{n}\left(\mathbf{X}_{i\cdot} - \boldsymbol{\mu}^{\mathrm{T}}\right)^{\mathrm{T}}\left(\mathbf{X}_{i\cdot} - \boldsymbol{\mu}^{\mathrm{T}}\right)$. In our example we have

$$\left(\mathbf{X}_{1\cdot} - \boldsymbol{\mu}^{\mathrm{T}}\right)^{\mathrm{T}}\left(\mathbf{X}_{1\cdot} - \boldsymbol{\mu}^{\mathrm{T}}\right) = \begin{pmatrix} 2 \\ -4 \end{pmatrix}\begin{pmatrix} 2 & -4 \end{pmatrix} = \begin{pmatrix} 4 & -8 \\ -8 & 16 \end{pmatrix}$$

$$\left(\mathbf{X}_{2\cdot} - \boldsymbol{\mu}^{\mathrm{T}}\right)^{\mathrm{T}}\left(\mathbf{X}_{2\cdot} - \boldsymbol{\mu}^{\mathrm{T}}\right) = \begin{pmatrix} 0 \\ 1 \end{pmatrix}\begin{pmatrix} 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$$

$$\left(\mathbf{X}_{3\cdot} - \boldsymbol{\mu}^{\mathrm{T}}\right)^{\mathrm{T}}\left(\mathbf{X}_{3\cdot} - \boldsymbol{\mu}^{\mathrm{T}}\right) = \begin{pmatrix} -2 \\ 3 \end{pmatrix}\begin{pmatrix} -2 & 3 \end{pmatrix} = \begin{pmatrix} 4 & -6 \\ -6 & 9 \end{pmatrix}$$

Background 7.2. Some more matrix notation.

Optimisation is a broad term denoting the problem of finding the best item or value among a set of alternatives. We have already seen a very simple, unconstrained form of optimisation in Example 7.1 on p.197, where we found the values of a and b minimising the sum of squared residuals $f(a,b) = \sum_{i=1}^{n}(w_i - (a + bh_i))^2$; this can be denoted as

$$a^*, b^* = \underset{a,b}{\arg\min}\, f(a,b)$$

f is called the *objective function*; it can be linear, quadratic (as in this case), or more complex. We found the minimum of f by setting the partial derivatives of f with respect to a and b to 0, and solving for a and b; the vector of these partial derivatives is called the *gradient* and denoted ∇f, so a succinct way of defining the unconstrained optimisation problem is: find a and b such that $\nabla f(a,b) = \mathbf{0}$. In this particular case the objective function is *convex*, which essentially means that there is a unique global minimum. This is, however, not always the case.

A *constrained optimisation* problem is one where the alternatives are subject to constraints, for instance

$$a^*, b^* = \underset{a,b}{\arg\min}\, f(a,b) \qquad \text{subject to } g(a,b) = c$$

If the relationship expressed by the constraint is linear, say $a - b = 0$, we can of course eliminate one of the variables and solve the simpler, unconstrained problem. However, this may not be possible if the constraints are non-linear. *Lagrange multipliers* are a powerful way of dealing with the general case. We form the Lagrange function defined by

$$\Lambda(a,b,\lambda) = f(a,b) - \lambda(g(a,b) - c)$$

where λ is the Lagrange multiplier, and solve the unconstrained problem $\nabla\Lambda(a,b,\lambda) = \mathbf{0}$. Since $\nabla_{a,b}\Lambda(a,b,\lambda) = \nabla f(a,b) - \lambda\nabla g(a,b)$ and $\nabla_{\lambda}\Lambda(a,b,\lambda) = g(a,b) - c$, this is a succinct way of requiring (*i*) that the gradients of f and g point in the same direction, and (*ii*) that the constraint is satisfied. We can include multiple equality constraints and also inequality constraints, each with their own Lagrange multiplier.

From the Lagrange function it is possible to derive a *dual* optimisation problem where we find the optimal values of the Lagrange multipliers. In general, the solution to the dual problem is only a lower bound on the solution to the *primal* problem, but under a set of conditions known as the *Karush–Kuhn–Tucker conditions* (*KKT*) the two solutions become equal. The quadratic optimisation problem posed by support vector machines is usually solved in its dual form.

Background 7.3. Basic concepts and terminology in mathematical optimisation.

7.6 Linear models: Summary and further reading

After considering logical models in the previous three chapters we had a good look at linear models in this chapter. Logical models are inherently non-numerical, and so deal with numerical features by using thresholds to convert them into two or more intervals. Linear models are almost diametrically opposite in that they can deal with numerical features directly but need to pre-process non-numerical features.[3] Geometrically, linear models use lines and planes to build the model, which essentially means that a certain increase or decrease in one of the features has the same effect, regardless of that feature's value or any of the other features. They are simple and robust to variations in the training data, but sometimes suffer from underfitting as a consequence.

☞ In Section 7.1 we considered the least-squares method that was originally conceived to solve a regression problem. This classical method, which derives its name from minimising the sum of squared residuals between predicted and actual function values, is described in innumerable introductory mathematics and engineering texts (and was one of the example programs I remember running on my father's Texas Instruments TI-58 programmable calculator). We first had a look at the problem in univariate form, and then derived the general solution as $\hat{\mathbf{w}} = (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y}$, where $(\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}$ is a transformation that decorrelates, centres and normalises the features. We then discussed regularised versions of linear regression: ridge regression was introduced by Hoerl and Kennard (1970), and the lasso which naturally leads to sparse solutions was introduced by Tibshirani (1996). We saw how the least-squares method could be applied to binary classification by encoding the classes by $+1$ and -1, leading to the solution $\hat{\mathbf{w}} = (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}(Pos\,\boldsymbol{\mu}^{\oplus} - Neg\,\boldsymbol{\mu}^{\ominus})$. This generalises the basic linear classifier by taking feature correlation and unequal class prevalence into account, but at a considerably increased computational cost (quadratic in the number of instances and cubic in the number of features).

☞ Section 7.2 presented another classical linear model, the perceptron. Unlike the least-squares method, which always finds the optimal solution in terms of sum of squared residuals, the perceptron is a heuristic algorithm that depends, for one thing, on the order in which the examples are presented. Invented by Rosenblatt (1958), its convergence for linearly separable data was proved by Novikoff (1962), who gave an upper bound on the number of mistakes made before the perceptron converged. Minsky and Papert (1969) proved further formal properties of the perceptron, but also demonstrated the limitations of a linear classifier. These were overcome with the development, over an extended period of time and with contributions from many people, of the multilayer perceptron and its

[3]Ways to pre-process non-numerical features for use in linear models are discussed in Chapter 10.

back-propagation training algorithm (Rumelhart, Hinton and Williams, 1986). In this section we also learned about the dual, instance-based view of linear classification in which we are learning instance weights rather than feature weights. For the perceptron these weights are the number of times the example has been misclassified during training.

☞ Maximum-margin classification with support vector machines was the topic of Section 7.3. The approach was proposed by Boser, Guyon and Vapnik (1992). Using the dual formulation, the instance weights are non-zero only for the support vectors, which are the training instances on the margin. The soft-margin generalisation is due to Cortes and Vapnik (1995). Margin errors are allowed, but the total margin error is added as a regularisation term to the objective function to be minimised, weighted by the complexity parameter C; all instances inside the margin receive instance weight C. As we have seen, by making C sufficiently small the support vector machine summarises the classes by their unweighted class means and hence is very similar to the basic linear classifier. A general introduction to SVMs is provided by Cristianini and Shawe-Taylor (2000). The sequential minimal optimisation algorithm is an often-used solver which iteratively selects pairs of multipliers to optimise analytically and is due to Platt (1998).

☞ In Section 7.4 we considered two methods to turn linear classifiers into probability estimators by converting the signed distance from the decision boundary into class probabilities. One well-known method is to use the logistic function, either straight out of the box or by fitting location and spread parameters to the data. Although this is often presented as a simple trick, we saw how it can be justified by assuming that the distances per class are normally distributed with the same variance; this latter assumption is needed to make the transformation monotonic. A non-parametric alternative is to use the ROC convex hull to obtain calibrated probability estimates. As was already mentioned in the summary of Chapter 2, the approach has its roots in isotonic regression (Best and Chakravarti, 1990) and was introduced to the machine learning community by Zadrozny and Elkan (2002). Fawcett and Niculescu-Mizil (2007) and Flach and Matsubara (2007) show its equivalence to calibration by means of the ROC convex hull.

☞ Finally, Section 7.5 discussed briefly how to go beyond linearity with kernel methods. The 'kernel trick' can be applied to any learning algorithm that can be entirely described in terms of dot products, which includes most approaches discussed in this chapter. The beauty is that we are implicitly classifying in a high-dimensional feature space, without having to construct the space explicitly. I gave the kernel perceptron as a simple example of a kernelised algorithm; in the next chapter we will see another example. Shawe-Taylor and Cristianini (2004) provide an excellent reference bringing together a wealth of material on the use of kernels in machine learning, and Gärtner (2009) discusses how kernel methods can be applied to structured, non-numerical data.

8.7 Distance-based models: Summary and further reading

Along with linear models, distance-based models are the second group of models with strong geometric intuitions. The literature on distance-based models is rich and diverse; in this chapter I've concentrated on getting the main intuitions across.

☞ In Section 8.1 we reviewed the most commonly used distance metrics: the Minkowski distance or p-norm with special cases Euclidean distance ($p = 2$) and Manhattan distance ($p = 1$); the Hamming distance, which counts the number of bits or literals that are different; and the Mahalanobis distance, which decorrelates and normalises the features (Mahalanobis, 1936). Other distances can be taken into account, as long as they satisfy the requirements of a distance metric listed in Definition 8.2.

☞ Section 8.2 investigated the key concepts of neighbours and exemplars. Exemplars are either centroids that find a centre of mass according to a chosen distance metric, or medoids that find the most centrally located data point. The most commonly used centroid is the arithmetic mean, which minimises squared Euclidean distance to all other points. Other definitions of centroids are possible but harder to compute: e.g., the geometric median is the point minimising Euclidean distance, but does not admit a closed-form solution. The complexity of finding a medoid is always quadratic regardless of the distance metric. We then considered nearest-neighbour decision rules, and looked in particular at the difference between 2-norm and 1-norm nearest-exemplar decision boundaries, and how these get refined by switching to a 2-nearest-exemplars decision rule.

☞ In Section 8.3 we discussed nearest-neighbour models which simply use the training data as exemplars. This is a very widely used model for classification, the origins of which can be traced back to Fix and Hodges (1951). Despite its simplicity, it can be shown that with sufficient training data the error rate is at most twice the optimal error rate (Cover and Hart, 1967). The 1-nearest neighbour classifier has low bias but high variance; by increasing the number of neighbours over which we aggregate we can reduce the variance but at the same time increase the bias. The nearest-neighbour decision rule can also be applied to real-valued target variables, and more generally to any task where we have an appropriate aggregator for multiple target values.

☞ Section 8.4 considered a number of algorithms for distance-based clustering using either arithmetic means or medoids. The K-means algorithm is a simple heuristic approach to solve the K-means problem that was originally proposed in 1957 and is sometimes referred to as Lloyd's algorithm (Lloyd, 1982). It is dependent on the initial configuration and can easily converge to the wrong stationary point. We also looked at the K-medoids and partitioning around medoids algorithms, the latter due to Kaufman and Rousseeuw (1990). These are computationally more expensive due to the use of medoids. Silhouettes (Rousseeuw, 1987) are a useful technique to check whether points are on average closer to the other members of their cluster than they are to the members of the neighbouring cluster. Much more detail about these and other clustering methods is provided by Jain, Murty and Flynn (1999).

☞ Whereas the previous clustering methods all result in a partition of the instance space and are therefore predictive, hierarchical clustering discussed in Section 8.5 applies only to the given data and is hence descriptive. A distinct advantage is that the clustering is constructed in the form of a dendrogram, which means that the number of clusters does not need to be specified in advance and can be chosen by inspecting the dendrogram. However, the method is computationally expensive and infeasible for large data sets. Furthermore, it is not always obvious which of the possible linkage functions to choose.

☞ Finally, in Section 8.6 we briefly considered how distances can be 'kernelised', and we gave one example in the form of kernel K-means. The use of a non-Euclidean distance metric leads to quadratic complexity of recalculating the clusters in each iteration.

The univariate normal or Gaussian distribution has the following probability density function:

$$P(x|\mu,\sigma) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) = \frac{1}{E}\exp\left(-\frac{1}{2}\left[\frac{x-\mu}{\sigma}\right]^2\right) = \frac{1}{E}\exp\left(-z^2/2\right), \quad E = \sqrt{2\pi}\sigma$$

The distribution has two parameters: μ, which is the mean or expected value, as well as the median (i.e., the point where the area under the density function is split in half) and the mode (i.e., the point where the density function reaches its maximum); and σ, which is the standard deviation and determines the width of the bell-shaped curve.

$z = (x-\mu)/\sigma$ is the *z-score* associated with x; it measures the number of standard deviations between x and the mean (it has itself mean 0 and standard deviation 1). It follows that $P(x|\mu,\sigma) = \frac{1}{\sigma}P(z|0,1)$, where $P(z|0,1)$ denotes the *standard normal distribution*. In other words, any normal distribution can be obtained from the standard normal distribution by scaling the x-axis with a factor σ, scaling the y-axis with a factor $1/\sigma$ (so the area under the curve remains 1), and translating the origin over μ.

The *multivariate normal distribution* over d-vectors $\mathbf{x} = (x_1,\ldots,x_d)^{\mathrm{T}} \in \mathbb{R}^d$ is

$$P(\mathbf{x}|\boldsymbol{\mu},\boldsymbol{\Sigma}) = \frac{1}{E_d}\exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right), \quad E_d = (2\pi)^{d/2}\sqrt{|\boldsymbol{\Sigma}|} \tag{9.2}$$

The parameters are the mean vector $\boldsymbol{\mu} = (\mu_1,\ldots,\mu_d)^{\mathrm{T}}$ and the d-by-d covariance matrix $\boldsymbol{\Sigma}$ (see Background 7.2 on p.200). $\boldsymbol{\Sigma}^{-1}$ is the inverse of the covariance matrix, and $|\boldsymbol{\Sigma}|$ is its determinant. The components of $\mathbf{x}$ may be thought of as d features that are possibly correlated.

If $d = 1$, then $\boldsymbol{\Sigma} = \sigma^2 = |\boldsymbol{\Sigma}|$ and $\boldsymbol{\Sigma}^{-1} = 1/\sigma^2$, which gives us the univariate Gaussian as a special case. For $d = 2$ we have $\boldsymbol{\Sigma} = \begin{pmatrix} \sigma_1^2 & \sigma_{12} \\ \sigma_{12} & \sigma_2^2 \end{pmatrix}$, $|\boldsymbol{\Sigma}| = \sigma_1^2\sigma_2^2 - (\sigma_{12})^2$ and $\boldsymbol{\Sigma}^{-1} = \frac{1}{|\boldsymbol{\Sigma}|}\begin{pmatrix} \sigma_2^2 & -\sigma_{12} \\ -\sigma_{12} & \sigma_1^2 \end{pmatrix}$. Using z-scores we derive the following expression for the bivariate normal distribution:

$$P(x_1,x_2|\mu_1,\mu_2,\sigma_1,\sigma_2,\rho) = \frac{1}{E_2}\exp\left(-\frac{1}{2(1-\rho^2)}(z_1^2+z_2^2-2\rho z_1 z_2)\right), \quad E_2 = 2\pi\sigma_1\sigma_2\sqrt{1-\rho^2} \tag{9.3}$$

where $z_i = (x_i-\mu_i)/\sigma_i$ for $i = 1,2$, and $\rho = \sigma_{12}/\sigma_1\sigma_2$ is the *correlation coefficient* between the two features.

The *multivariate standard normal distribution* has $\boldsymbol{\mu} = \mathbf{0}$ (a d-vector with all 0s) and $\boldsymbol{\Sigma} = \mathbf{I}$ (the d-by-d identity matrix), and thus $P(\mathbf{x}|\mathbf{0},\mathbf{I}) = \frac{1}{(2\pi)^{d/2}}\exp\left(-\frac{1}{2}\mathbf{x}\cdot\mathbf{x}\right)$.

Background 9.1. The normal distribution.

The *Bernoulli distribution*, named after the Swiss seventeenth century mathematician Jacob Bernoulli, concerns Boolean or binary events with two possible outcomes: success or 1, and failure or 0. A Bernoulli distribution has a single parameter θ which gives the probability of success: hence $P(X=1)=\theta$ and $P(X=0)=1-\theta$. The Bernoulli distribution has expected value $\mathbb{E}[X]=\theta$ and variance $\mathbb{E}\left[(X-\mathbb{E}[X])^2\right]=\theta(1-\theta)$.

The *binomial distribution* arises when counting the number of successes S in n independent Bernoulli trials with the same parameter θ. It is described by

$$P(S=s)=\binom{n}{s}\theta^s(1-\theta)^{n-s} \text{ for } s\in\{0,\dots,n\}$$

This distribution has expected value $\mathbb{E}[S]=n\theta$ and variance $\mathbb{E}\left[(S-\mathbb{E}[S])^2\right]=n\theta(1-\theta)$.

The *categorical distribution* generalises the Bernoulli distribution to $k\geq 2$ outcomes. The parameter of the distribution is a k-vector $\boldsymbol{\theta}=\left(\theta_1,\dots,\theta_k\right)$ such that $\sum_{i=1}^k\theta_i=1$.

Finally, the *multinomial distribution* tabulates the outcomes of n independent and identically distributed (i.i.d.) categorical trials. That is, $\mathbf{X}=\left(X_1,\dots,X_k\right)$ is a k-vector of integer counts, and

$$P(\mathbf{X}=\left(x_1,\dots,x_k\right))=n!\frac{\theta_1^{x_1}}{x_1!}\cdots\frac{\theta_k^{x_k}}{x_k!}$$

with $\sum_{i=1}^k x_i=n$. Notice that setting $n=1$ gives us an alternative way of stating the categorical distribution as $P(\mathbf{X}=\left(x_1,\dots,x_k\right))=\theta_1^{x_1}\cdots\theta_k^{x_k}$, with exactly one of the x_i equal to 1 and the rest set to 0. Furthermore, setting $k=2$ gives an alternative expression for the Bernoulli distribution as $P(X=x)=\theta^x(1-\theta)^{1-x}$ for $x\in\{0,1\}$. It is also useful to note that if $\mathbf{X}$ follows a multinomial distribution, then each component X_i follows a binomial distribution with parameter θ_i.

We can estimate the parameters of these distributions by counting in a straightforward way. Suppose $a\ b\ a\ c\ c\ b\ a\ a\ b\ c$ is a sequence of words. We might be interested in individual words being a or not, and interpret the data as coming from 10 i.i.d. Bernoulli trials, which would allow us to estimate $\hat{\theta}_a=4/10=0.4$. This same parameter generates a binomial distribution of the number of occurrences of the word a in similar sequences. Alternatively, we can estimate the parameters of the categorical (word occurrences) and multinomial (word counts) distributions as $\hat{\boldsymbol{\theta}}=(0.4,0.3,0.3)$.

It is almost always a good idea to smooth these distributions by including *pseudo-counts*. Imagine our vocabulary includes the word d but we haven't yet observed it, then a maximum-likelihood estimate would set $\hat{\theta}_d=0$. We can smooth this by adding a virtual occurrence of each word to our observations, leading to $\hat{\boldsymbol{\theta}}'=(5/14,4/14,4/14,1/14)$. In the case of a binomial this is the Laplace correction.

Background 9.2. Probability distributions for categorical data.

perspective, where we need to define a prior distribution on models. The MDL viewpoint offers a concrete way of defining model priors by means of codes.

9.6 Probabilistic models: Summary and further reading

In this chapter we covered a range of machine learning models that are all based on the idea that features and target variables can be modelled as random variables, giving the opportunity to explicitly represent and manipulate the level of certainty we have about those variables. Such models are usually predictive in that they result in a conditional distribution $P(Y|X)$ with which Y can be predicted from X. Generative models estimate the joint distribution $P(Y,X)$ – often through the likelihood $P(X|Y)$ and the prior $P(Y)$ – from which the posterior $P(Y|X)$ can be obtained, while conditional models learn the posterior $P(Y|X)$ directly without spending resources on learning $P(X)$. The 'Bayesian' approach to machine learning is characterised by concentrating on the full posterior distribution wherever this is feasible, rather than just deriving a maximising value.

☞ In Section 9.1 we saw that the normal or Gaussian distribution supports many useful geometric intuitions, essentially because the negative logarithm of the Gaussian likelihood can be interpreted as a squared distance. Straight decision boundaries result from having the same per-class covariance matrices, which means that models resulting in such linear boundaries, including linear classifiers, linear regression and K-means clustering, can be interpreted from a probabilistic viewpoint that makes their inherent assumptions explicit. Two examples of this are that the basic linear classifier is Bayes-optimal for uncorrelated, unit-variance Gaussian features; and least-squares regression is optimal for linear functions contaminated by Gaussian noise on the target variable.

☞ Section 9.2 was devoted to different versions of the naive Bayes classifier, which makes the simplifying assumption that features are independent within each class. Lewis (1998) gives an overview and history. This model is widely used in information retrieval and text classification as it is often a good ranker if not a good probability estimator. While the model that is usually understood as naive Bayes treats features as categorical or Bernoulli random variables, variants employing a multinomial model tend to better model the number of occurrences of words in a document (McCallum and Nigam, 1998). Real-valued features can be taken into account by either modelling them as normally distributed within each class, or by non-parametric density estimation – John and Langley (1995) suggest that the latter gives better empirical results. Webb, Boughton and Wang (2005) discuss ways of relaxing the strong independence assumptions made by naive Bayes. Probability smoothing by means of the m-estimate was introduced by Cestnik (1990).

☞ Perhaps paradoxically, I don't think there is anything particularly 'Bayesian' about the naive Bayes classifier. While it is a generative probabilistic model estimating the posterior $P(Y|X)$ through the joint $P(Y,X)$, in practice the posterior is very poorly calibrated owing to the unrealistic independence assumptions. The reason naive Bayes is often successful is because of the quality of $\arg\max_Y P(Y|X)$ rather than the quality of the posterior as such, as analysed by Domingos and Pazzani (1997). Furthermore, even the use of Bayes' rule in determining the maximising Y can be avoided, as it only serves to transform uncalibrated likelihoods into uncalibrated posteriors. So my recommendation is to use naive Bayes likelihoods as scores on an unknown scale whose decision threshold needs to be calibrated by means of ROC analysis, as has been discussed several times before.

☞ In Section 9.3 we looked at the widely used logistic regression model. The basic idea is to combine a linear decision boundary with logistic calibration, but to train this in a discriminative fashion by optimising conditional likelihood. So, rather than modelling the classes as clouds of points and deriving a decision boundary from those clouds, logistic regression concentrates on areas of class overlap. It is an instance of the larger class of generalised linear models (Nelder and Wedderburn, 1972). Jebara (2004) discusses the advantages of discriminative learning in comparison with generative models. Discriminative learning can also be applied to sequential data in the form of conditional random fields (Lafferty *et al.*, 2001)

☞ Section 9.4 presented the Expectation-Maximisation algorithm as a general way of learning models involving unobserved variables. This general form of EM was proposed by Dempster, Laird and Rubin (1977) based on a variety of earlier work. We have seen how it can be applied to Gaussian mixture models to obtain a more general version of K-means predictive clustering, which is also able to estimate cluster shapes and sizes. However, this increases the number of parameters of the model and thus the risk of getting stuck in a non-optimal stationary configuration. (Little and Rubin, 1987) is a standard reference for dealing with missing data.

☞ Finally, in Section 9.5 we briefly discussed some ideas related to learning as compression. The link with probabilistic modelling is that both seek to model and exploit the non-random aspects of the data. In a simplified setting, the minimum description length principle can be derived from Bayes' rule by taking the negative logarithm, and states that models minimising the description length of the model and of the data given the model should be preferred. The first term quantifies the complexity of the model, and the second term quantifies its accuracy (as only the model's errors need to be encoded explicitly). The advantage of the MDL principle is that encoding schemes are often more tangible and easier to define than prior distributions. However, not just any encoding will do: as with their probabilistic counterparts, these schemes need to be justified in the domain being modelled. Pioneering work in this area has been done by Solomonoff (1964*a*,*b*); Wallace and Boulton (1968); Rissanen (1978), among others. An excellent introduction and overview is provided by Grünwald (2007).

Imagine a swimmer who swims the same distance d on two different days, taking a seconds one day and b seconds the next. On average, it took her therefore $c = (a+b)/2$ seconds, with an average speed of $d/c = 2d/(a+b)$. Notice how this average speed is *not* calculated as the normal or *arithmetic mean* of the speeds, which would yield $(d/a+d/b)/2$: to calculate average speed over a fixed distance we use a different mean called the *harmonic mean*. Given two numbers x and y (in our swimming example these are the speeds on either day, d/a and d/b), the harmonic mean h is defined as

$$h(x,y) = \frac{2}{1/x+1/y} = \frac{2xy}{x+y}$$

Since $1/h(x,y) = (1/x+1/y)/2$, we observe that calculating the harmonic mean on a scale with unit u corresponds to calculating the arithmetic mean on the *reciprocal scale* with unit $1/u$. In the example, speed with fixed distance is expressed on a scale reciprocal to the time scale, and since we use the arithmetic mean to average time, we use the harmonic mean to average speed. (If we average speed over a fixed *time* interval this is expressed on the same scale as distance and thus we would use the arithmetic mean.)

A good example of where the harmonic mean is used in machine learning arises when we average precision and recall of a classifier. Remember that precision is the proportion of positive predictions that is correct ($prec = TP/(TP+FP)$), and recall is the proportion of positives that is correctly predicted ($rec = TP/(TP+FN)$). Suppose we first calculate the number of mistakes averaged over the classes: this is the arithmetic mean $Fm = (FP+FN)/2$. We can then derive

$$\frac{TP}{TP+Fm} = \frac{TP}{TP+(FP+FN)/2} = \frac{2TP}{(TP+FP)+(TP+FN)} = \frac{2}{1/prec+1/rec}$$

We recognise the last term as the harmonic mean of precision and recall. Since the enumerator of both precision and recall is fixed, taking the arithmetic mean of the denominators corresponds to taking the harmonic mean of the ratios. In information retrieval this harmonic mean of precision and recall is very often used and called the *F-measure*.

Yet other means exist for other scales. In music, going from one note to a note one octave higher corresponds to doubling the frequency. So frequencies f and $4f$ are two octaves apart, and it makes sense to take the octave in between with frequency $2f$ as their mean. This is achieved by the *geometric mean*, which is defined as $g(x,y) = \sqrt{xy}$. Since $\log\sqrt{xy} = (\log xy)/2 = (\log x + \log y)/2$ it follows that the geometric mean corresponds to the arithmetic mean on a logarithmic scale. All these means have in common that the mean of two values is an intermediate value, and that they can easily be extended to more than two values.

Background 10.1. On scales and means.

name *latent semantic indexing* (*LSA*) ('latent' is synonymous with 'hidden'). Instead of film genres, LSA uncovers document topics by decomposing matrices containing word counts per document, under the assumption that the word counts per topic are independent and can thus simply be added up.[3] The other main application of matrix factorisation is *completion* of missing entries in a matrix, the idea being that if we approximate the observed entries in the matrix as closely as possible using a low-rank decomposition, this allows us to infer the missing entries.

10.4 Features: Summary and further reading

In this chapter we have given features some long-overdue attention. Features are the telescopes through which we observe the data universe and therefore an important unifying force in machine learning. Features are related to measurements in science, but there is no widespread consensus on how to formalise and categorise different measurements – I have taken inspiration from Stevens' scales of measurements (Stevens, 1946), but otherwise aimed to stay close to current practice in machine learning.

- ☞ The main kinds of feature distinguished in Section 10.1 are categorical, ordinal and quantitative features. The latter are expressed on a quantitative scale and admit the calculation of the widest range of statistics of tendency (mean, median, mode; see (von Hippel, 2005) for a discussion of rules of thumb regarding these), dispersion (variance and standard deviation, range, interquartile range) and shape (skewness and kurtosis). In machine learning quantitative features are often referred to as continuous features, but I think this term is inappropriate as it wrongly suggests that their defining feature is somehow an unlimited precision. It is important to realise that quantitative features do not necessarily have an additive scale: e.g., quantitative features expressing a probability are expressed on a multiplicative scale, and the use of Euclidean distance, say, would be inappropriate for non-additive features. Ordinal features have order but not scale; and categorical features (also called nominal or discrete) have neither order nor scale.

- ☞ Structured features are first-order logical statements that refer to parts of objects by means of local variables and use some kind of aggregation, such as existential quantification or counting, to extract a property of the main object. Constructing first-order features prior to learning is often referred to as propositionalisation;

[3]Other models are possible: e.g., in Boolean matrix decomposition the matrix product is changed to a Boolean product in which integer addition is replaced by Boolean disjunction (so that $1+1=1$), with the effect that additional topics do not provide additional explanatory power for the occurrence of a word in a document.

Kramer *et al.* (2000) and Lachiche (2010) give surveys, and an experimental comparison of different approaches is carried out by Krogel *et al.* (2003).

☞ In Section 10.2 we looked at a number of feature transformations. Discretisation and thresholding are the best-known of these, turning a quantitative feature into a categorical or a Boolean one. One of the most effective discretisation methods is the recursive partitioning algorithm using information gain to find the thresholds and a stopping criterion derived from the minimum description length principle proposed by Fayyad and Irani (1993). Other overviews and proposals are given by Boullé (2004, 2006). The agglomerative merging approach using χ^2 was proposed by Kerber (1992).

☞ We have seen that in a two-class setting, supervised discretisation can be visualised by means of coverage curves. This then naturally leads to the idea of using these coverage curves and their convex hull to calibrate rather than just discretise the features. After all, ordinal and quantitative features are univariate rankers and scoring classifiers and thus the same classifier calibration methods can be applied, in particular logistic and isotonic calibration as discussed in Section 7.4. The calibrated features live in probability space, but we might prefer to work with log-odds space instead as this is additive rather than multiplicative. Fitting data to a fixed linear decision boundary in calibrated log-odds space is closely related to training a naive Bayes model. Isotonic calibration leads to piecewise axis-parallel decision boundaries; owing to the discretising nature of isotonic calibration this can be understood as the constructing of a grouping model, even if the original model in the uncalibrated space was a grading model.

☞ Section 10.3 was devoted to feature construction and selection. Early approaches to feature construction and constructive induction were proposed by Ragavan and Rendell (1993); Donoho and Rendell (1995). The instance-based Relief feature selection method is due to Kira and Rendell (1992) and extended by Robnik-Sikonja and Kononenko (2003). The distinction between filter approaches to feature selection – which evaluate features on their individual merits – and wrapper approaches, which evaluate sets of features, is originally due to Kohavi and John (1997). Hall (1999) proposes a filter approach called correlation-based feature selection that aims at combining the best of both worlds. Guyon and Elisseeff (2003) give an excellent introduction to feature selection.

☞ Finally, we looked at feature construction and selection from a linear algebra perspective. Matrix decomposition and factorisation is an actively researched technique that was instrumental in winning a recent film recommendation challenge worth $1 million (Koren *et al.*, 2009). Decomposition techniques employing additional constraints include non-negative matrix decomposition (Lee *et al.*, 1999).

Boolean matrix decomposition is studied by Miettinen (2009). Mahoney and Drineas (2009) describe a matrix decomposition technique that uses actual columns and rows of the data matrix to preserve sparsity (unlike SVD which produces dense matrices even if the original matrix is sparse). Latent semantic indexing and a probabilistic extension is described by Hofmann (1999). Ding and He (2004) discuss the relationship between K-means clustering and principal component analysis.

performance on a particular data set. It follows that we can only hope to achieve useful meta-learning over non-uniform distributions of learning problems.

11.4 Model ensembles: Summary and further reading

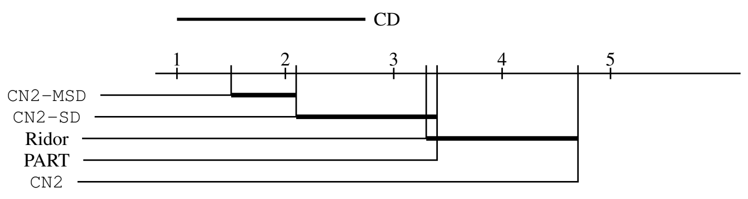
In this short chapter we have discussed some of the fundamental ideas underlying ensemble methods. What all ensemble methods have in common is that they construct several base models from adapted versions of the training data, on top of which some technique is employed to combine the predictions or scores from the base models into a single prediction of the ensemble. We focused on bagging and boosting as two of the most commonly used ensemble methods. A good introduction to model ensembles is given by Brown (2010). The standard reference on classifier combination is Kuncheva (2004) and a more recent overview is given by Zhou (2012).

☞ In Section 11.1 we discussed bagging and random forests. Bagging trains diverse models from samples of the training data, and was introduced by Breiman (1996*a*). Random forests, usually attributed to Breiman (2001), combine bagged decision trees with random subspaces; similar ideas were developed by Ho (1995) and Amit and Geman (1997). These techniques are particularly useful to reduce the variance of low-bias models such as tree models.

☞ Boosting was discussed in Section 11.2. The key idea is to train diverse models by increasing the weight of previously misclassified examples. This helps to reduce the bias of otherwise stable learners such as linear classifiers or decision stumps. An accessible overview is given by Schapire (2003). Kearns and Valiant (1989, 1994) posed the question whether a weak learning algorithm that performs just slightly better than random guessing can be boosted into an arbitrarily accurate strong learning algorithm. Schapire (1990) introduced a theoretical form of boosting to show the equivalence of weak and strong learnability. The AdaBoost algorithm on which Algorithm 11.3 is based was introduced by Freund and Schapire (1997). Schapire and Singer (1999) give multi-class and multi-label extensions of AdaBoost. A ranking version of AdaBoost was proposed by Freund *et al.* (2003). The boosted rule learning approach that can handle classifiers that may abstain was inspired by Slipper (Cohen and Singer, 1999), a boosted version of Ripper (Cohen, 1995).

☞ In Section 11.3 we discussed bagging and boosting in terms of bias and variance. Schapire, Freund, Bartlett and Lee (1998) provide a detailed theoretical and experimental analysis of boosting in terms of improving the margin distribution. I also mentioned some other ensemble methods that train a meta-model for combining the base models. Stacking employs a linear meta-model and was introduced by Wolpert (1992) for classification and extended by Breiman (1996*b*) for regression. Meta-decision trees were introduced by Todorovski and Dzeroski (2003).

☞ We also briefly discussed meta-learning as a technique for learning about the performance of learning algorithms. The field originated from an early empirical study documented by Michie *et al.* (1994). Recent references are Brazdil *et al.* (2009, 2010). Unpruned and unpruned decision trees were used to obtain data set characteristics by Peng *et al.* (2002). The idea of training simple models to obtain further data characteristics is known as landmarking (Pfahringer *et al.*, 2000).



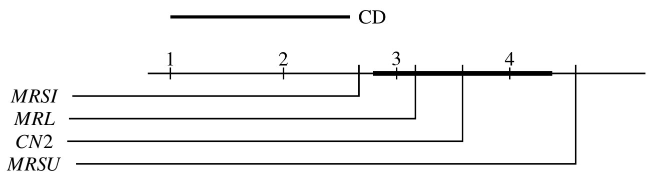


Figure 12.1. (top) Critical difference diagram for the pairwise Nemenyi test. Average ranks for each algorithm are plotted on the real axis. The critical difference is shown as a bar above the figure, and any group of consecutively ranked algorithms such that the outermost ones are less than the critical difference apart are connected by a horizontal thick line. The diagram shows, e.g., that the performance of the top ranked algorithm is significantly better than the bottom three. **(bottom)** Critical difference diagram for the Bonferroni–Dunn test with CN2 as control. The critical differences are now drawn symmetrically around the average rank of the control. The top ranked algorithm is significantly better than the control, and the bottom ranked one is significantly worse. (Figures courtesy of Tarek Abudawood (2011)).

A variant of the Nemenyi test called the *Bonferroni–Dunn test* can be applied when we perform pairwise tests only against a control algorithm. The calculation of the critical difference is the same, except q_α is adjusted to reflect the fact that we make $k-1$ pairwise comparisons rather than $k(k-1)/2$. For example, for $\alpha = 0.05$ and $k = 3$ we have $q_\alpha = 2.241$, which is slightly lower than the value used for the Nemenyi test, leading to a tighter critical difference. Figure 12.1 (bottom) shows a graphical representation of the Bonferroni–Dunn post-hoc test.

12.4 Machine learning experiments: Summary and further reading

In this chapter we have taken a look at how we can use data to answer questions about the performance of models and learning algorithms. A 'machine learning experimenter' needs to address three questions: (*i*) what to measure, (*ii*) how to measure it, and (*iii*) how to interpret it. An excellent source – particularly for the last two questions – is Japkowicz and Shah (2011).

☞ In order to decide what to measure, we first need to explicate our experimental objective. We also need to consider the operating context: performance aspects that might change when using the model. For example, the operating context might be given by the class distribution, but we may have no prior knowledge telling us that certain distributions are more likely than others. Example 12.1 on p.345 demonstrated that in such a case average recall would be more appropriate as a performance measure, even if the experimental objective is accuracy. We also looked at the difference between a precision–recall analysis which ignores the true negatives, and a true/false positive rate analysis which takes them into account; a fuller analysis is provided by Davis and Goadrich (2006). The relation between accuracy as experimental objective and AUC as performance measure is studied by Hernández-Orallo *et al.* (2011).

☞ Once we decided what to measure, we need to establish a measuring protocol. The most common approach is k-fold cross-validation, which divides the data into k folds, repeatedly trains on $k-1$ of those and tests on the remaining one. It is of paramount importance that there be no information leak between the training data used to learn the model and the test data used to evaluate it. A common mistake is to use cross-validation to find the best setting of one or more parameters of a learning algorithm, say the complexity parameter of a support vector machine. This is methodologically wrong as parameter tuning should be carried out as part of the training process, without any access to the test data. A methodologically sound option is to use *internal cross-validation* by setting aside a validation fold in each cross-validation run for parameter tuning. Experimental studies regarding cross-validation are carried out by Dietterich (1998) and Bouckaert and Frank (2004): the former recommends five times two-fold cross-validation and the latter ten times ten-fold. ROC curves can be drawn in cross-validation as each instance appears in a test fold exactly once, and so we can collect the scores on all test folds. Fawcett (2006) considers alternatives including horizontal and vertical averaging.

☞ In the context of interpreting experimental results we looked at confidence intervals and significance tests. Confidence intervals have a clear statistical interpretation: they quantify the likelihood of a measurement falling in a particular interval, assuming a particular true value. Significance tests extend this to reasoning about a particular null hypothesis, such as ‘these learning algorithms do not perform differently on these data sets’. Significance tests are designed for particular protocols: the t-test can be used for evaluating two learning algorithms on two data sets, Wilcoxon’s signed-rank test is applicable for comparing two algorithms over multiple data sets, and Friedman’s test (or analysis of variance) compares multiple algorithms over multiple data sets. An excellent discussion of these and related tests is provided by Demšar (2006).

☞ It should be mentioned that there is much discussion on the use of significance tests in machine learning, and on the wider issue regarding machine learning as an experimental science. The importance of experiments in machine learning was stressed early on by Pat Langley in two influential papers (Langley, 1988; Kibler and Langley, 1988); however, more recently he expressed criticism at the way experimental methodology in machine learning has become rather inflexible (Langley, 2011). Other authors critical of current practice include Drummond (2006) and Demšar (2008).

Epilogue: Where to go from here

AND SO WE HAVE come to the end of our journey through the 'making sense of data' landscape. We have seen how machine learning can build models from features for solving tasks involving data. We have seen how models can be predictive or descriptive; learning can be supervised or unsupervised; and models can be logical, geometric, probabilistic or ensembles of such models. Now that I have equipped you with the basic concepts to understand the literature, there is a whole world out there for you to explore. So it is only natural for me to leave you with a few pointers to areas you may want to learn about next.

One thing that we have often assumed in the book is that the data comes in a form suitable for the task at hand. For example, if the task is to label e-mails we conveniently learn a classifier from data in the form of labelled e-mails. For tasks such as class probability estimation I introduced the output space (for the model) as separate from the label space (for the data) because the model outputs (class probability estimates) are not directly observable in the data and have to be reconstructed. An area where the distinction between data and model output is much more pronounced is *reinforcement learning*. Imagine you want to learn how to be a good chess player. This could be viewed as a classification task, but then you require a teacher to score every move. What happens in practical situations is that every now and then you receive a reward or a punishment – e.g., winning the game, or losing one of your pieces. The challenge is then to assign credit or blame to individual moves that led to such rewards or punishments being incurred. Reinforcement learning is a principled way to learn policies for deciding which action to take in which situation or state. This is currently one of the most active subfields of machine learning. The standard reference is Sutton and Barto (1998), and you should have no trouble finding more recent workshop proceedings or journal special issues.

There are many other tasks that require us to relax some of our assumptions. For example, in multi-class classification we assume that classes are mutually exclusive. In *multi-label classification* we drop that assumption, so that an instance can be labelled with an arbitrary subset of labels. This is natural, e.g., when tagging online material such as blog posts. The dependence between labels is an additional source of information: for example, knowing that the tag 'machine learning' applies makes the tag 'reinforcement learning' more likely. Multi-label learning aims to exploit this information by learning the dependence between the labels as well as the mapping between the features and each individual label. For relevant work in the area see, e.g., Tsoumakas *et al.* (2012). A related area is *preference learning*, where the goal is to learn instance-dependent preferences between class labels (Fürnkranz and Hüllermeier, 2010). Increasing the complexity of the model outputs even further, we arrive at the general area of *structured output prediction* (Bakir *et al.*, 2007).

Going back to multi-label learning, although each label establishes a separate binary classification task, the goal is to avoid learning completely separate models for each task. This is, in fact, a special case of what is called *multi-task learning*. For example, each task could be to predict a separate real-valued target variable on the same instance space, and the learner is aiming to exploit, say, correlations between the target variables. Closely related to this is the area of *transfer learning*, which studies the transfer of models between tasks. A relevant reference for both areas is Silver and Bennett (2008).

Another assumption that deserves closer scrutiny is the availability of data in a single batch. In *online learning*, also called incremental learning, the model needs to be updated each time a new data point arrives. One application of this is in the area of *sequence prediction* (Cesa-Bianchi and Lugosi, 2006). With the increase in sensor data this setting is rapidly gaining importance, as can be witnessed from the growing area of *learning from data streams* (Gama and Gaber, 2007). Sometimes it is convenient to give the learner a more active role in data acquisition, for example by issuing queries for examples to be labelled by the teacher. *Active learning* studies exactly this setting (Settles, 2011).

Ultimately, machine learning is – and, in all likelihood, will remain – a research area at the nexus of two distinct developments. On the one hand, it is widely recognised that the ability for learning and self-training is necessary for achieving machine intelligence in any form. An area in machine learning that has this quest at heart is *deep learning*, which aims at employing hierarchies of autonomously constructed features (Bengio, 2009). On the other hand, machine learning is an indispensable tool for dealing with the data deluge. Building machine learning models is an essential step in the *data mining* process, which poses specific challenges such as being able to deal with ‘big data’ and cloud computing platforms. I hope that this book has kindled your interest in one of these exciting developments.

Important points to remember

By decreasing a model's refinement we sometimes achieve better ranking performance. 69

Concavities in ROC curves can be remedied by combining segments through tied scores. 77

To avoid overfitting, the number of parameters estimated from the data must be considerably less than the number of data points. 93

In descriptive learning the task and learning problem coincide. 95

The LGG is the most conservative generalisation that we can learn from the data. 108

Every concept between the least general one and one of the most general ones is also a possible hypothesis. 112

An upward path through the hypothesis space corresponds to a coverage curve. 114

Decision trees are strictly more expressive than conjunctive concepts. 131

One way to avoid overfitting and encourage learning is to deliberately choose a restrictive hypothesis language. 131

The ranking obtained from the empirical probabilities in the leaves of a decision tree yields a convex ROC curve on the training data. 138

Entropy and Gini index are sensitive to fluctuations in the class distribution, $\sqrt{\text{Gini}}$ isn't. 147

Rule lists are similar to decision trees in that the empirical probabilities associated with each rule yield convex ROC and coverage curves on the training data. 166

$(\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}$ acts as a transformation that decorrelates, centres and normalises the features. 202

Assuming uncorrelated features effectively decomposes a multivariate regression problem into d univariate problems. 203

A general way of constructing a linear classifier with decision boundary $\mathbf{w}\cdot\mathbf{x} = t$ is by constructing $\mathbf{w}$ as $\mathbf{M}^{-1}(n^{\oplus}\boldsymbol{\mu}^{\oplus} - n^{\ominus}\boldsymbol{\mu}^{\ominus})$. 207

In the dual, instance-based view of linear classification we are learning instance weights α_i rather than feature weights w_j. 210

A minimal-complexity soft margin classifier summarises the classes by their class means in a way very similar to the basic linear classifier. 219

The basic linear classifier can be interpreted from a distance-based perspective as constructing exemplars that minimise squared Euclidean distance within each class, and then applying a nearest-exemplar decision rule. 239

References

Abudawood, T. (**2011**). Multi-class subgroup discovery: Heuristics, algorithms and predictiveness. Ph.D. thesis, University of Bristol, Department of Computer Science, Faculty of Engineering. 357

Abudawood, T. and Flach, P.A. (**2009**). Evaluation measures for multi-class subgroup discovery. In W.L. Buntine, M. Grobelnik, D. Mladenić and J. Shawe-Taylor (eds.), *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases (ECML-PKDD 2009), Part I*, *LNCS*, volume 5781, pp. 35–50. Springer. 193

Agrawal, R., Imielinski, T. and Swami, A.N. (**1993**). Mining association rules between sets of items in large databases. In P. Buneman and S. Jajodia (eds.), *Proceedings of the ACM International Conference on Management of Data (SIGMOD 1993)*, pp. 207–216. ACM Press. 103

Agrawal, R., Mannila, H., Srikant, R., Toivonen, H. and Verkamo, A.I. (**1996**). Fast discovery of association rules. In *Advances in Knowledge Discovery and Data Mining*, pp. 307–328. AAAI/MIT Press. 193

Allwein, E.L., Schapire, R.E. and Singer, Y. (**2000**). Reducing multiclass to binary: A unifying approach for margin classifiers. In P. Langley (ed.), *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000)*, pp. 9–16. Morgan Kaufmann. 102

Amit, **Y. and Geman**, **D.** (**1997**). Shape quantization and recognition with randomized trees. *Neural Computation* 9(7):1545–1588. 341

Angluin, **D.**, **Frazier**, **M. and Pitt**, **L.** (**1992**). Learning conjunctions of Horn clauses. *Machine Learning* 9:147–164. 128

Bakir, **G.**, **Hofmann**, **T.**, **Schölkopf**, **B.**, **Smola**, **A.J.**, **Taskar**, **B. and Vishwanathan**, **S.V.N.** (**2007**). *Predicting Structured Data*. MIT Press. 361

Banerji, **R.B.** (**1980**). *Artificial Intelligence: A Theoretical Approach*. Elsevier Science. 127

Bengio, **Y.** (**2009**). Learning deep architectures for AI. *Foundations and Trends in Machine Learning* 2(1):1–127. 361

Best, **M.J. and Chakravarti**, **N.** (**1990**). Active set algorithms for isotonic regression; a unifying framework. *Mathematical Programming* 47(1):425–439. 80, 229

Blockeel, **H.** (**2010***a*). Hypothesis language. In C. Sammut and G.I. Webb (eds.), *Encyclopedia of Machine Learning*, pp. 507–511. Springer. 127

Blockeel, **H.** (**2010***b*). Hypothesis space. In C. Sammut and G.I. Webb (eds.), *Encyclopedia of Machine Learning*, pp. 511–513. Springer. 127

Blockeel, **H.**, **De Raedt**, **L. and Ramon**, **J.** (**1998**). Top-down induction of clustering trees. In J.W. Shavlik (ed.), *Proceedings of the Fifteenth International Conference on Machine Learning (ICML 1998)*, pp. 55–63. Morgan Kaufmann. 103, 156

Blumer, **A.**, **Ehrenfeucht**, **A.**, **Haussler**, **D. and Warmuth**, **M.K.** (**1989**). Learnability and the Vapnik-Chervonenkis dimension. *Journal of the ACM* 36(4):929–965. 128

Boser, **B.E.**, **Guyon**, **I. and Vapnik**, **V.** (**1992**). A training algorithm for optimal margin classifiers. In *Proceedings of the International Conference on Computational Learning Theory (COLT 1992)*, pp. 144–152. 229

Bouckaert, **R. and Frank**, **E.** (**2004**). Evaluating the replicability of significance tests for comparing learning algorithms. In H. Dai, R. Srikant and C. Zhang (eds.), *Advances in Knowledge Discovery and Data Mining, LNCS*, volume 3056, pp. 3–12. Springer. 358

Boullé, **M.** (**2004**). Khiops: A statistical discretization method of continuous attributes. *Machine Learning* 55(1):53–69. 328

Boullé, **M.** (**2006**). MODL: A Bayes optimal discretization method for continuous attributes. *Machine Learning* 65(1):131–165. 328

Bourke, C., Deng, K., Scott, S.D., Schapire, R.E. and Vinodchandran, N.V. (**2008**). On reoptimizing multi-class classifiers. *Machine Learning* 71(2-3):219–242. 102

Brazdil, P., Giraud-Carrier, C.G., Soares, C. and Vilalta, R. (**2009**). *Metalearning – Applications to Data Mining*. Springer. 342

Brazdil, P., Vilalta, R., Giraud-Carrier, C.G. and Soares, C. (**2010**). Metalearning. In C. Sammut and G.I. Webb (eds.), *Encyclopedia of Machine Learning*, pp. 662–666. Springer. 342

Breiman, L. (**1996***a*). Bagging predictors. *Machine Learning* 24(2):123–140. 341

Breiman, L. (**1996***b*). Stacked regressions. *Machine Learning* 24(1):49–64. 342

Breiman, L. (**2001**). Random forests. *Machine Learning* 45(1):5–32. 341

Breiman, L., Friedman, J.H., Olshen, R.A. and Stone, C.J. (**1984**). *Classification and Regression Trees*. Wadsworth. 156

Brier, G.W. (**1950**). Verification of forecasts expressed in terms of probability. *Monthly Weather Review* 78(1):1–3. 80

Brown, G. (**2010**). Ensemble learning. In C. Sammut and G.I. Webb (eds.), *Encyclopedia of Machine Learning*, pp. 312–320. Springer. 341

Bruner, J.S., Goodnow, J.J. and Austin, G.A. (**1956**). *A Study of Thinking*. Science Editions. 2nd edn 1986. 127

Cesa-Bianchi, N. and Lugosi, G. (**2006**). *Prediction, Learning, and Games*. Cambridge University Press. 361

Cestnik, B. (**1990**). Estimating probabilities: A crucial task in machine learning. In *Proceedings of the European Conference on Artificial Intelligence (ECAI 1990)*, pp. 147–149. 296

Clark, P. and Boswell, R. (**1991**). Rule induction with CN2: Some recent improvements. In Y. Kodratoff (ed.), *Proceedings of the European Working Session on Learning (EWSL 1991), LNCS*, volume 482, pp. 151–163. Springer. 192

Clark, P. and Niblett, T. (**1989**). The CN2 induction algorithm. *Machine Learning* 3:261–283. 192

Cohen, W.W. (**1995**). Fast effective rule induction. In A. Prieditis and S.J. Russell (eds.), *Proceedings of the Twelfth International Conference on Machine Learning (ICML 1995)*, pp. 115–123. Morgan Kaufmann. 192, 341

Cohen, **W.W. and Singer**, **Y.** (**1999**). A simple, fast, and efficive rule learner. In J. Hendler and D. Subramanian (eds.), *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI 1999)*, pp. 335–342. AAAI Press / MIT Press. 341

Cohn, **D.** (**2010**). Active learning. In C. Sammut and G.I. Webb (eds.), *Encyclopedia of Machine Learning*, pp. 10–14. Springer. 128

Cortes, **C. and Vapnik**, **V.** (**1995**). Support-vector networks. *Machine Learning* 20(3):273–297. 229

Cover, **T. and Hart**, **P.** (**1967**). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory* 13(1):21–27. 260

Cristianini, **N. and Shawe-Taylor**, **J.** (**2000**). *An Introduction to Support Vector Machines*. Cambridge University Press. 229

Dasgupta, **S.** (**2010**). Active learning theory. In C. Sammut and G.I. Webb (eds.), *Encyclopedia of Machine Learning*, pp. 14–19. Springer. 128

Davis, **J. and Goadrich**, **M.** (**2006**). The relationship between precision-recall and ROC curves. In W.W. Cohen and A. Moore (eds.), *Proceedings of the Twenty-Third International Conference on Machine Learning (ICML 2006)*, pp. 233–240. ACM Press. 358

De Raedt, **L.** (**1997**). Logical settings for concept-learning. *Artificial Intelligence* 95(1):187–201. 128

De Raedt, **L.** (**2008**). *Logical and Relational Learning*. Springer. 193

De Raedt, **L.** (**2010**). Logic of generality. In C. Sammut and G.I. Webb (eds.), *Encyclopedia of Machine Learning*, pp. 624–631. Springer. 128

De Raedt, **L. and Kersting**, **K.** (**2010**). Statistical relational learning. In C. Sammut and G.I. Webb (eds.), *Encyclopedia of Machine Learning*, pp. 916–924. Springer. 193

Dempster, **A.P.**, **Laird**, **N.M. and Rubin**, **D.B.** (**1977**). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B (Methodological)* pp. 1–38. 296

Demšar, **J.** (**2008**). On the appropriateness of statistical tests in machine learning. In *Proceedings of the ICML'08 Workshop on Evaluation Methods for Machine Learning*. 359

Demšar, **J.** (**2006**). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research* 7:1–30. 359

Dietterich, T.G. (1998). Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation* 10(7):1895–1923. 358

Dietterich, T.G. and Bakiri, G. (1995). Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research* 2:263–286. 102

Dietterich, T.G., Kearns, M.J. and Mansour, Y. (1996). Applying the weak learning framework to understand and improve c4.5. In *Proceedings of the Thirteenth International Conference on Machine Learning*, pp. 96–104. 156

Ding, C.H.Q. and He, X. (2004). *K*-means clustering via principal component analysis. In C.E. Brodley (ed.), *Proceedings of the Twenty-First International Conference on Machine Learning (ICML 2004)*. ACM Press. 329

Domingos, P. and Pazzani, M. (1997). On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning* 29(2):103–130. 296

Donoho, S.K. and Rendell, L.A. (1995). Rerepresenting and restructuring domain theories: A constructive induction approach. *Journal of Artificial Intelligence Research* 2:411–446. 328

Drummond, C. (2006). Machine learning as an experimental science (revisited). In *Proceedings of the AAAI'06 Workshop on Evaluation Methods for Machine Learning*. 359

Drummond, C. and Holte, R.C. (2000). Exploiting the cost (in)sensitivity of decision tree splitting criteria. In P. Langley (ed.), *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000)*, pp. 239–246. Morgan Kaufmann. 156

Egan, J.P. (1975). *Signal Detection Theory and ROC Analysis*. Academic Press. 80

Fawcett, T. (2006). An introduction to ROC analysis. *Pattern Recognition Letters* 27(8):861–874. 80, 358

Fawcett, T. and Niculescu-Mizil, A. (2007). PAV and the ROC convex hull. *Machine Learning* 68(1):97–106. 80, 229

Fayyad, U.M. and Irani, K.B. (1993). Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI 1993)*, pp. 1022–1029. 328

Ferri, C., Flach, P.A. and Hernández-Orallo, J. (2002). Learning decision trees using the area under the ROC curve. In C. Sammut and A.G. Hoffmann (eds.), *Proceedings of the Nineteenth International Conference on Machine Learning (ICML 2002)*, pp. 139–146. Morgan Kaufmann. 156

Ferri, **C.**, **Flach**, **P.A. and Hernández-Orallo**, **J.** (**2003**). Improving the AUC of probabilistic estimation trees. In N. Lavrač, D. Gamberger, L. Todorovski and H. Blockeel (eds.), *Proceedings of the European Conference on Machine Learning (ECML 2003), LNCS*, volume 2837, pp. 121–132. Springer. 156

Fix, **E. and Hodges**, **J.L.** (**1951**). Discriminatory analysis. Nonparametric discrimination: Consistency properties. Technical report, USAF School of Aviation Medicine, Texas: Randolph Field. Report Number 4, Project Number 21-49-004. 260

Flach, **P.A.** (**1994**). *Simply Logical – Intelligent Reasoning by Example*. Wiley. 193

Flach, **P.A.** (**2003**). The geometry of ROC space: Understanding machine learning metrics through ROC isometrics. In T. Fawcett and N. Mishra (eds.), *Proceedings of the Twentieth International Conference on Machine Learning (ICML 2003)*, pp. 194–201. AAAI Press. 156

Flach, **P.A.** (**2010***a*). First-order logic. In C. Sammut and G.I. Webb (eds.), *Encyclopedia of Machine Learning*, pp. 410–415. Springer. 128

Flach, **P.A.** (**2010***b*). ROC analysis. In C. Sammut and G.I. Webb (eds.), *Encyclopedia of Machine Learning*, pp. 869–875. Springer. 80

Flach, **P.A. and Lachiche**, **N.** (**2001**). Confirmation-guided discovery of first-order rules with Tertius. *Machine Learning* 42(1/2):61–95. 193

Flach, **P.A. and Matsubara**, **E.T.** (**2007**). A simple lexicographic ranker and probability estimator. In J.N. Kok, J. Koronacki, R.L. de Mántaras, S. Matwin, D. Mladenic and A. Skowron (eds.), *Proceedings of the Eighteenth European Conference on Machine Learning (ECML 2007), LNCS*, volume 4701, pp. 575–582. Springer. 80, 229

Freund, **Y.**, **Iyer**, **R.D.**, **Schapire**, **R.E. and Singer**, **Y.** (**2003**). An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research* 4:933–969. 341

Freund, **Y. and Schapire**, **R.E.** (**1997**). A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci.* 55(1):119–139. 341

Fürnkranz, **J.** (**1999**). Separate-and-conquer rule learning. *Artificial Intelligence Review* 13(1):3–54. 192

Fürnkranz, **J.** (**2010**). Rule learning. In C. Sammut and G.I. Webb (eds.), *Encyclopedia of Machine Learning*, pp. 875–879. Springer. 192

Fürnkranz, **J. and Flach**, **P.A.** (**2003**). An analysis of rule evaluation metrics. In T. Fawcett and N. Mishra (eds.), *Proceedings of the Twentieth International Conference on Machine Learning (ICML 2003)*, pp. 202–209. AAAI Press. 79

Fürnkranz, **J. and Flach**, **P.A.** (**2005**). ROC 'n' Rule learning – towards a better understanding of covering algorithms. *Machine Learning* 58(1):39–77. 192

Fürnkranz, **J.**, **Gamberger**, **D. and Lavrač**, **N.** (**2012**). *Foundations of Rule Learning*. Springer. 192

Fürnkranz, **J. and Hüllermeier**, **E.** (eds.) (**2010**). *Preference Learning*. Springer. 361

Fürnkranz, **J. and Widmer**, **G.** (**1994**). Incremental reduced error pruning. In *Proceedings of the Eleventh International Conference on Machine Learning (ICML 1994)*, pp. 70–77. 192

Gama, **J. and Gaber**, **M.M.** (eds.) (**2007**). *Learning from Data Streams: Processing Techniques in Sensor Networks*. Springer. 361

Ganter, **B. and Wille**, **R.** (**1999**). *Formal Concept Analysis: Mathematical Foundations*. Springer. 127

Garriga, **G.C.**, **Kralj**, **P. and Lavrač**, **N.** (**2008**). Closed sets for labeled data. *Journal of Machine Learning Research* 9:559–580. 127

Gärtner, **T.** (**2009**). *Kernels for Structured Data*. World Scientific. 230

Grünwald, **P.D.** (**2007**). *The Minimum Description Length Principle*. MIT Press. 297

Guyon, **I. and Elisseeff**, **A.** (**2003**). An introduction to variable and feature selection. *Journal of Machine Learning Research* 3:1157–1182. 328

Hall, **M.A.** (**1999**). Correlation-based feature selection for machine learning. Ph.D. thesis, University of Waikato. 328

Han, **J.**, **Cheng**, **H.**, **Xin**, **D. and Yan**, **X.** (**2007**). Frequent pattern mining: Current status and future directions. *Data Mining and Knowledge Discovery* 15(1):55–86. 193

Hand, **D.J. and Till**, **R.J.** (**2001**). A simple generalisation of the area under the ROC curve for multiple class classification problems. *Machine Learning* 45(2):171–186. 102

Haussler, **D.** (**1988**). Quantifying inductive bias: AI learning algorithms and Valiant's learning framework. *Artificial Intelligence* 36(2):177–221. 128

Hernández-Orallo, **J.**, **Flach**, **P.A. and Ferri**, **C.** (**2011**). Threshold choice methods: The missing link. Available online at `http://arxiv.org/abs/1112.2640`. 358

Ho, **T.K.** (**1995**). Random decision forests. In *Proceedings of the International Conference on Document Analysis and Recognition*, p. 278. IEEE Computer Society, Los Alamitos, CA, USA. 341

Hoerl, A.E. and Kennard, R.W. (**1970**). Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics* pp. 55–67. 228

Hofmann, T. (**1999**). Probabilistic latent semantic indexing. In *Proceedings of the Twenty-Second Annual International ACM Conference on Research and Development in Information Retrieval (SIGIR 1999)*, pp. 50–57. ACM Press. 329

Hunt, E.B., Marin, J. and Stone, P.J. (**1966**). *Experiments in Induction.* Academic Press. 127, 156

Jain, A.K., Murty, M.N. and Flynn, P.J. (**1999**). Data clustering: A review. *ACM Computing Surveys* 31(3):264–323. 261

Japkowicz, N. and Shah, M. (**2011**). *Evaluating Learning Algorithms: A Classification Perspective.* Cambridge University Press. 357

Jebara, T. (**2004**). *Machine Learning: Discriminative and Generative.* Springer. 296

John, G.H. and Langley, P. (**1995**). Estimating continuous distributions in Bayesian classifiers. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence (UAI 1995)*, pp. 338–345. Morgan Kaufmann. 295

Kaufman, L. and Rousseeuw, P.J. (**1990**). *Finding Groups in Data: An Introduction to Cluster Analysis.* John Wiley. 261

Kearns, M.J. and Valiant, L.G. (**1989**). Cryptographic limitations on learning Boolean formulae and finite automata. In D.S. Johnson (ed.), *Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing (STOC 1989)*, pp. 433–444. ACM Press. 341

Kearns, M.J. and Valiant, L.G. (**1994**). Cryptographic limitations on learning Boolean formulae and finite automata. *Journal of the ACM* 41(1):67–95. 341

Kerber, R. (**1992**). Chimerge: Discretization of numeric attributes. In *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI 1992)*, pp. 123–128. AAAI Press. 328

Kibler, D.F. and Langley, P. (**1988**). Machine learning as an experimental science. In *Proceedings of the European Working Session on Learning (EWSL 1988)*, pp. 81–92. 359

King, R.D., Srinivasan, A. and Dehaspe, L. (**2001**). Warmr: A data mining tool for chemical data. *Journal of Computer-Aided Molecular Design* 15(2):173–181. 193

Kira, **K. and Rendell**, **L.A.** (**1992**). The feature selection problem: Traditional methods and a new algorithm. In W.R. Swartout (ed.), *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI 1992)*, pp. 129–134. AAAI Press / MIT Press. 328

Klösgen, **W.** (**1996**). Explora: A multipattern and multistrategy discovery assistant. In *Advances in Knowledge Discovery and Data Mining*, pp. 249–271. MIT Press. 103

Kohavi, **R. and John**, **G.H.** (**1997**). Wrappers for feature subset selection. *Artificial Intelligence* 97(1-2):273–324. 328

Koren, **Y.**, **Bell**, **R. and Volinsky**, **C.** (**2009**). Matrix factorization techniques for recommender systems. *IEEE Computer* 42(8):30–37. 328

Kramer, **S.** (**1996**). Structural regression trees. In *Proceedings of the National Conference on Artificial Intelligence (AAAI 1996)*, pp. 812–819. 156

Kramer, **S.**, **Lavrač**, **N. and Flach**, **P.A.** (**2000**). Propositionalization approaches to relational data mining. In S. Džeroski and N. Lavrač (eds.), *Relational Data Mining*, pp. 262–286. Springer. 328

Krogel, **M.A.**, **Rawles**, **S.**, **Zelezný**, **F.**, **Flach**, **P.A.**, **Lavrač**, **N. and Wrobel**, **S.** (**2003**). Comparative evaluation of approaches to propositionalization. In T. Horváth (ed.), *Proceedings of the Thirteenth International Conference on Inductive Logic Programming (ILP 2003)*, *LNCS*, volume 2835, pp. 197–214. Springer. 328

Kuncheva, **L.I.** (**2004**). *Combining Pattern Classifiers: Methods and Algorithms*. John Wiley and Sons. 341

Lachiche, **N.** (**2010**). Propositionalization. In C. Sammut and G.I. Webb (eds.), *Encyclopedia of Machine Learning*, pp. 812–817. Springer. 328

Lachiche, **N. and Flach**, **P.A.** (**2003**). Improving accuracy and cost of two-class and multi-class probabilistic classifiers using ROC curves. In T. Fawcett and N. Mishra (eds.), *Proceedings of the Twentieth International Conference on Machine Learning (ICML 2003)*, pp. 416–423. AAAI Press. 102

Lafferty, **J.D.**, **McCallum**, **A. and Pereira**, **F.C.N.** (**2001**). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In C.E. Brodley and A.P. Danyluk (eds.), *Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001)*, pp. 282–289. Morgan Kaufmann. 296

Langley, **P.** (**1988**). Machine learning as an experimental science. *Machine Learning* 3:5–8. 359

Langley, **P.** (**1994**). *Elements of Machine Learning*. Morgan Kaufmann. 156

Langley, **P.** (**2011**). The changing science of machine learning. *Machine Learning* 82(3):275–279. 359

Lavrač, **N.**, **Kavšek**, **B.**, **Flach**, **P.A. and Todorovski**, **L.** (**2004**). Subgroup discovery with CN2-SD. *Journal of Machine Learning Research* 5:153–188. 193

Lee, **D.D.**, **Seung**, **H.S.** *et al.* (**1999**). Learning the parts of objects by non-negative matrix factorization. *Nature* 401(6755):788–791. 328

Leman, **D.**, **Feelders**, **A. and Knobbe**, **A.J.** (**2008**). Exceptional model mining. In W. Daelemans, B. Goethals and K. Morik (eds.), *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases (ECML-PKDD 2008), Part II*, *LNCS*, volume 5212, pp. 1–16. Springer. 103

Lewis, **D.** (**1998**). Naive Bayes at forty: The independence assumption in information retrieval. In *Proceedings of the Tenth European Conference on Machine Learning (ECML 1998)*, pp. 4–15. Springer. 295

Li, **W.**, **Han**, **J. and Pei**, **J.** (**2001**). CMAR: Accurate and efficient classification based on multiple class-association rules. In N. Cercone, T.Y. Lin and X. Wu (eds.), *Proceedings of the IEEE International Conference on Data Mining (ICDM 2001)*, pp. 369–376. IEEE Computer Society. 193

Little, **R.J.A. and Rubin**, **D.B.** (**1987**). *Statistical Analysis with Missing Data*. Wiley. 296

Liu, **B.**, **Hsu**, **W. and Ma**, **Y.** (**1998**). Integrating classification and association rule mining. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD 1998)*, pp. 80–86. AAAI Press. 193

Lloyd, **J.W.** (**2003**). *Logic for Learning – Learning Comprehensible Theories from Structured Data*. Springer. 193

Lloyd, **S.** (**1982**). Least squares quantization in PCM. *IEEE Transactions on Information Theory* 28(2):129–137. 261

Mahalanobis, **P.C.** (**1936**). On the generalised distance in statistics. *Proceedings of the National Institute of Science, India* 2(1):49–55. 260

Mahoney, **M.W. and Drineas**, **P.** (**2009**). CUR matrix decompositions for improved data analysis. *Proceedings of the National Academy of Sciences* 106(3):697. 329

McCallum, **A. and Nigam**, **K.** (**1998**). A comparison of event models for naive Bayes text classification. In *Proceedings of the AAAI-98 Workshop on Learning for Text Categorization*, pp. 41–48. 295

Michalski, **R.S.** (**1973**). Discovering classification rules using variable-valued logic system VL_1. In *Proceedings of the Third International Joint Conference on Artificial Intelligence*, pp. 162–172. Morgan Kaufmann Publishers. 127

Michalski, **R.S.** (**1975**). Synthesis of optimal and quasi-optimal variable-valued logic formulas. In *Proceedings of the 1975 International Symposium on Multiple-Valued Logic*, pp. 76–87. 192

Michie, **D.**, **Spiegelhalter**, **D.J. and Taylor**, **C.C.** (**1994**). *Machine Learning, Neural and Statistical Classification.* Ellis Horwood. 342

Miettinen, **P.** (**2009**). Matrix decomposition methods for data mining: Computational complexity and algorithms. Ph.D. thesis, University of Helsinki. 329

Minsky, **M. and Papert**, **S.** (**1969**). *Perceptrons: An Introduction to Computational Geometry.* MIT Press. 228

Mitchell, **T.M.** (**1977**). Version spaces: A candidate elimination approach to rule learning. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, pp. 305–310. Morgan Kaufmann Publishers. 127

Mitchell, **T.M.** (**1997**). *Machine Learning.* McGraw-Hill. 128

Muggleton, **S.** (**1995**). Inverse entailment and Progol. *New Generation Computing* 13(3&4):245–286. 193

Muggleton, **S.**, **De Raedt**, **L.**, **Poole**, **D.**, **Bratko**, **I.**, **Flach**, **P.A.**, **Inoue**, **K. and Srinivasan**, **A.** (**2012**). ILP turns 20 – biography and future challenges. *Machine Learning* 86(1):3–23. 193

Muggleton, **S. and Feng**, **C.** (**1990**). Efficient induction of logic programs. In *Proceedings of the International Conference on Algorithmic Learning Theory (ALT 1990)*, pp. 368–381. 193

Murphy, **A.H. and Winkler**, **R.L.** (**1984**). Probability forecasting in meteorology. *Journal of the American Statistical Association* pp. 489–500. 80

Nelder, **J.A. and Wedderburn**, **R.W.M.** (**1972**). Generalized linear models. *Journal of the Royal Statistical Society, Series A (General)* pp. 370–384. 296

Novikoff, **A.B.** (**1962**). On convergence proofs on perceptrons. In *Proceedings of the Symposium on the Mathematical Theory of Automata*, volume 12, pp. 615–622. Polytechnic Institute of Brooklyn, New York. 228

Pasquier, N., Bastide, Y., Taouil, R. and Lakhal, L. (1999). Discovering frequent closed itemsets for association rules. In *Proceedings of the International Conference on Database Theory (ICDT 1999)*, pp. 398–416. Springer. 127

Peng, Y., Flach, P.A., Soares, C. and Brazdil, P. (2002). Improved dataset characterisation for meta-learning. In S. Lange, K. Satoh and C.H. Smith (eds.), *Proceedings of the Fifth International Conference on Discovery Science (DS 2002)*, *LNCS*, volume 2534, pp. 141–152. Springer. 342

Pfahringer, B., Bensusan, H. and Giraud-Carrier, C.G. (2000). Meta-learning by landmarking various learning algorithms. In P. Langley (ed.), *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000)*, pp. 743–750. Morgan Kaufmann. 342

Platt, J.C. (1998). Using analytic QP and sparseness to speed training of support vector machines. In M.J. Kearns, S.A. Solla and D.A. Cohn (eds.), *Advances in Neural Information Processing Systems 11 (NIPS 1998)*, pp. 557–563. MIT Press. 229

Plotkin, G.D. (1971). Automatic methods of inductive inference. Ph.D. thesis, University of Edinburgh. 127

Provost, F.J. and Domingos, P. (2003). Tree induction for probability-based ranking. *Machine Learning* 52(3):199–215. 156

Provost, F.J. and Fawcett, T. (2001). Robust classification for imprecise environments. *Machine Learning* 42(3):203–231. 79

Quinlan, J.R. (1986). Induction of decision trees. *Machine Learning* 1(1):81–106. 155

Quinlan, J.R. (1990). Learning logical definitions from relations. *Machine Learning* 5:239–266. 193

Quinlan, J.R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann. 156

Ragavan, H. and Rendell, L.A. (1993). Lookahead feature construction for learning hard concepts. In *Proceedings of the Tenth International Conference on Machine Learning (ICML 1993)*, pp. 252–259. Morgan Kaufmann. 328

Rajnarayan, D.G. and Wolpert, D. (2010). Bias-variance trade-offs: Novel applications. In C. Sammut and G.I. Webb (eds.), *Encyclopedia of Machine Learning*, pp. 101–110. Springer. 103

Rissanen, J. (1978). Modeling by shortest data description. *Automatica* 14(5):465–471. 297

Rivest, R.L. (1987). Learning decision lists. *Machine Learning* 2(3):229–246. 192

Robnik-Sikonja, **M. and Kononenko**, **I.** (**2003**). Theoretical and empirical analysis of ReliefF and RReliefF. *Machine Learning* 53(1-2):23–69. 328

Rosenblatt, **F.** (**1958**). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review* 65(6):386. 228

Rousseeuw, **P.J.** (**1987**). Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics* 20(0):53–65. 261

Rumelhart, **D.E.**, **Hinton**, **G.E. and Williams**, **R.J.** (**1986**). Learning representations by back-propagating errors. *Nature* 323(6088):533–536. 229

Schapire, **R.E.** (**1990**). The strength of weak learnability. *Machine Learning* 5:197–227. 341

Schapire, **R.E.** (**2003**). The boosting approach to machine learning: An overview. In *Nonlinear Estimation and Classification*, pp. 149–172. Springer. 341

Schapire, **R.E.**, **Freund**, **Y.**, **Bartlett**, **P. and Lee**, **W.S.** (**1998**). Boosting the margin: A new explanation for the effectiveness of voting methods. *Annals of Statistics* 26(5):1651–1686. 341

Schapire, **R.E. and Singer**, **Y.** (**1999**). Improved boosting algorithms using confidence-rated predictions. *Machine Learning* 37(3):297–336. 341

Settles, **B.** (**2011**). *Active Learning*. Morgan & Claypool. 361

Shawe-Taylor, **J. and Cristianini**, **N.** (**2004**). *Kernel Methods for Pattern Analysis*. Cambridge University Press. 230

Shotton, **J.**, **Fitzgibbon**, **A.W.**, **Cook**, **M.**, **Sharp**, **T.**, **Finocchio**, **M.**, **Moore**, **R.**, **Kipman**, **A. and Blake**, **A.** (**2011**). Real-time human pose recognition in parts from single depth images. In *Proceedings of the Twenty-Fourth IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2011)*, pp. 1297–1304. 155

Silver, **D. and Bennett**, **K.** (**2008**). Guest editor's introduction: special issue on inductive transfer learning. *Machine Learning* 73(3):215–220. 361

Solomonoff, **R.J.** (**1964*a***). A formal theory of inductive inference: Part I. *Information and Control* 7(1):1–22. 297

Solomonoff, **R.J.** (**1964*b***). A formal theory of inductive inference: Part II. *Information and Control* 7(2):224–254. 297

Srinivasan, **A.** (**2007**). The Aleph manual, version 4 and above. Available online at `www.cs.ox.ac.uk/activities/machlearn/Aleph/`. 193

Stevens, **S.S.** (**1946**). On the theory of scales of measurement. *Science* 103(2684):677–680. 327

Sutton, **R.S. and Barto**, **A.G.** (**1998**). *Reinforcement Learning: An Introduction.* MIT Press. 361

Tibshirani, **R.** (**1996**). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B (Methodological)* pp. 267–288. 228

Todorovski, **L. and Dzeroski**, **S.** (**2003**). Combining classifiers with meta decision trees. *Machine Learning* 50(3):223–249. 342

Tsoumakas, **G.**, **Zhang**, **M.L. and Zhou**, **Z.H.** (**2012**). Introduction to the special issue on learning from multi-label data. *Machine Learning* 88(1-2):1–4. 361

Tukey, **J.W.** (**1977**). *Exploratory Data Analysis.* Addison-Wesley. 103

Valiant, **L.G.** (**1984**). A theory of the learnable. *Communications of the ACM* 27(11):1134–1142. 128

Vapnik, **V.N. and Chervonenkis**, **A.Y.** (**1971**). On uniform convergence of the frequencies of events to their probabilities. *Teoriya Veroyatnostei I Ee Primeneniya* 16(2):264–279. 128

Vere, **S.A.** (**1975**). Induction of concepts in the predicate calculus. In *Proceedings of the Fourth International Joint Conference on Artificial Intelligence*, pp. 281–287. 127

von Hippel, **P.T.** (**2005**). Mean, median, and skew: Correcting a textbook rule. *Journal of Statistics Education* 13(2). 327

Wallace, **C.S. and Boulton**, **D.M.** (**1968**). An information measure for classification. *Computer Journal* 11(2):185–194. 297

Webb, **G.I.** (**1995**). Opus: An efficient admissible algorithm for unordered search. *Journal of Artificial Intelligence Research* 3:431–465. 192

Webb, **G.I.**, **Boughton**, **J.R. and Wang**, **Z.** (**2005**). Not so naive Bayes: Aggregating one-dependence estimators. *Machine Learning* 58(1):5–24. 295

Winston, **P.H.** (**1970**). Learning structural descriptions from examples. Technical report, MIT Artificial Intelligence Lab. AITR-231. 127

Wojtusiak, **J.**, **Michalski**, **R.S.**, **Kaufman**, **K.A. and Pietrzykowski**, **J.** (**2006**). The AQ21 natural induction program for pattern discovery: Initial version and its novel features. In *Proceedings of the Eighteenth IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2006)*, pp. 523–526. 192

Wolpert, **D.H.** (**1992**). Stacked generalization. *Neural Networks* 5(2):241–259. 342

Zadrozny, **B. and Elkan**, **C.** (**2002**). Transforming classifier scores into accurate multiclass probability estimates. In *Proceedings of the Eighth ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD 2002)*, pp. 694–699. ACM Press. 80, 229

Zeugmann, **T.** (**2010**). PAC learning. In C. Sammut and G.I. Webb (eds.), *Encyclopedia of Machine Learning*, pp. 745–753. Springer. 128

Zhou, **Z.H.** (**2012**). *Ensemble Methods: Foundations and Algorithms.* Taylor & Francis. 341

Index