Demystifying Machine Learning

A Jean Golding Institute workshop

Peter Flach and Niall Twomey

Intelligent Systems Laboratory, University of Bristol, United Kingdom

December 5, 2017

Part two

Example 6.3, p.167

Learning a rule set for class $\oplus$

The first rule learned for the positive class is

$$\cdot \textbf{if} \; \text{Length} = 3 \; \textbf{then} \; \text{Class} = \oplus \cdot$$

The two examples covered by this rule are removed, and a new rule is learned. We now encounter a new situation, as none of the candidates is pure. We thus start a second-level search, from which the following pure rule emerges:

$$\cdot \textbf{if} \; \text{Gills} = \text{no} \wedge \text{Length} = 5 \; \textbf{then} \; \text{Class} = \oplus \cdot$$

To cover the remaining positive, we again need a rule with two conditions:

$$\cdot \textbf{if} \; \text{Gills} = \text{no} \wedge \text{Teeth} = \text{many} \; \textbf{then} \; \text{Class} = \oplus \cdot$$

What's next?

Example 7.1, p.197

Univariate linear regression

Suppose we want to investigate the relationship between people's height and weight. We collect n height and weight measurements $(h_i, w_i), 1 \leq i \leq n$.

Univariate linear regression assumes a linear equation $w = a + bh$, with parameters a and b chosen such that the sum of squared residuals $\sum_{i=1}^{n}(w_i - (a + bh_i))^2$ is minimised.

In order to find the parameters we take partial derivatives of this expression, set the partial derivatives to 0 and solve for a and b:

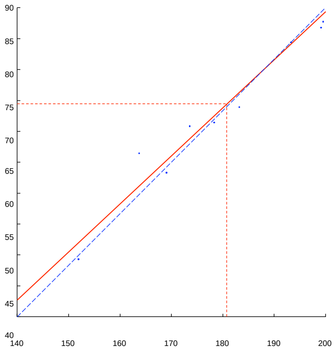
$$\frac{\partial}{\partial a}\sum_{i=1}^{n}(w_i - (a + bh_i))^2 = -2\sum_{i=1}^{n}(w_i - (a + bh_i)) = 0 \qquad \Rightarrow \hat{a} = \overline{w} - \hat{b}\overline{h}$$

$$\frac{\partial}{\partial b}\sum_{i=1}^{n}(w_i - (a + bh_i))^2 = -2\sum_{i=1}^{n}(w_i - (a + bh_i))h_i = 0 \qquad \Rightarrow \hat{b} = \frac{\sum_{i=1}^{n}(h_i - \overline{h})(w_i - \overline{w})}{\sum_{i=1}^{n}(h_i - \overline{h})^2}$$

So the solution found by linear regression is $w = \hat{a} + \hat{b}h = \overline{w} + \hat{b}(h - \overline{h})$; see Figure 7.1 for an example.

Figure 7.1, p.197

Univariate linear regression



The red solid line indicates the result of applying linear regression to 10 measurements of body weight (on the y-axis, in kilograms) against body height (on the x-axis, in centimetres). The orange dotted lines indicate the average height $\overline{h} = 181$ and the average weight $\overline{w} = 74.5$; the regression coefficient $\hat{b} = 0.78$. The measurements were simulated by adding normally distributed noise with mean 0 and variance 5 to the true model indicated by the blue dashed line ($b = 0.83$).

The perceptron

A linear classifier that will achieve perfect separation on linearly separable data is the *perceptron*, originally proposed as a simple neural network. The perceptron iterates over the training set, updating the weight vector every time it encounters an incorrectly classified example.

- For example, let $\mathbf{x}_i$ be a misclassified positive example, then we have $y_i = +1$ and $\mathbf{w} \cdot \mathbf{x}_i < t$. We therefore want to find $\mathbf{w}^f$ such that $\mathbf{w}^f \cdot \mathbf{x}_i > \mathbf{w} \cdot \mathbf{x}_i$, which moves the decision boundary towards and hopefully past x_i.
- This can be achieved by calculating the new weight vector as $\mathbf{w}^f = \mathbf{w} + \eta \mathbf{x}_i$, where $0 < \eta \leq 1$ is the *learning rate* (often set to 1). We then have $\mathbf{w}^f \cdot \mathbf{x}_i = \mathbf{w} \cdot \mathbf{x}_i + \eta \mathbf{x}_i \cdot \mathbf{x}_i > \mathbf{w} \cdot \mathbf{x}_i$ as required.
- Similarly, if $\mathbf{x}_j$ is a misclassified negative example, then we have $y_j = -1$ and $\mathbf{w} \cdot \mathbf{x}_j > t$. In this case we calculate the new weight vector as $\mathbf{w}^f = \mathbf{w} - \eta \mathbf{x}_j$, and thus $\mathbf{w}^f \cdot \mathbf{x}_j = \mathbf{w} \cdot \mathbf{x}_j - \eta \mathbf{x}_j \cdot \mathbf{x}_j < \mathbf{w} \cdot \mathbf{x}_j$.
- The two cases can be combined in a single update rule:

$$\mathbf{w}^f = \mathbf{w} + \eta y_i \mathbf{x}_i$$

Linear classifiers in dual form

Every time an example $\mathbf{x}_i$ is misclassified, we add $y_i \mathbf{x}_i$ to the weight vector.

- After training has completed, each example has been misclassified zero or more times. Denoting this number as α_i for example $\mathbf{x}_i$, the weight vector can be expressed as

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$$

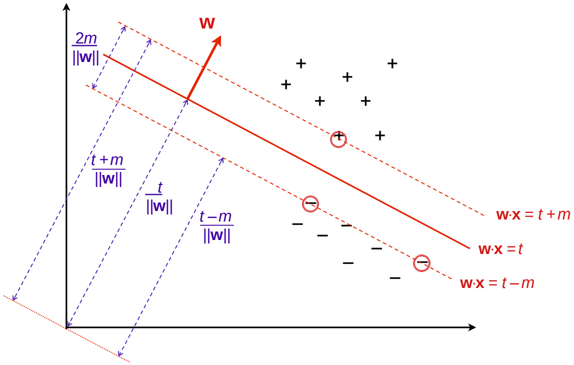
- In the dual, instance-based view of linear classification we are learning instance weights α_i rather than feature weights w_j. An instance $\mathbf{x}$ is classified as

$$\hat{y} = \text{sign}\left(\sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i \cdot \mathbf{x}\right)$$

- During training, the only information needed about the training data is all pairwise dot products: the n-by-n matrix $\mathbf{G} = \mathbf{X}\mathbf{X}^{\mathrm{T}}$ containing these dot products is called the *Gram matrix*.

Figure 7.7, p.212

Support vector machine



w
$\frac{2m}{||\mathbf{w}||}$
$\frac{t+m}{||\mathbf{w}||}$
$\frac{t}{||\mathbf{w}||}$
$\frac{t-m}{||\mathbf{w}||}$
$\mathbf{w}\cdot\mathbf{x} = t+m$
$\mathbf{w}\cdot\mathbf{x} = t$
$\mathbf{w}\cdot\mathbf{x} = t-m$

The geometry of a support vector classifier. The circled data points are the support vectors, which are the training examples nearest to the decision boundary. The support vector machine finds the decision boundary that maximises the margin $m/||\mathbf{w}||$.

Maximising the margin

Since we are free to rescale t, $||\mathbf{w}||$ and m, it is customary to choose $m = 1$. Maximising the margin then corresponds to minimising $||\mathbf{w}||$ or, more conveniently, $\frac{1}{2}||\mathbf{w}||^2$, provided of course that none of the training points fall inside the margin.

This leads to a quadratic, constrained optimisation problem:

$$\mathbf{w}^*, t^* = \arg\min_{\mathbf{w},t} \frac{1}{2}||\mathbf{w}||^2 \qquad \text{subject to } y_i(\mathbf{w}\cdot\mathbf{x}_i - t) \geq 1, 1 \leq i \leq n$$

Using the method of Lagrange multipliers, the dual form of this problem can be derived (seeBackground 7.3).

SVM in dual form

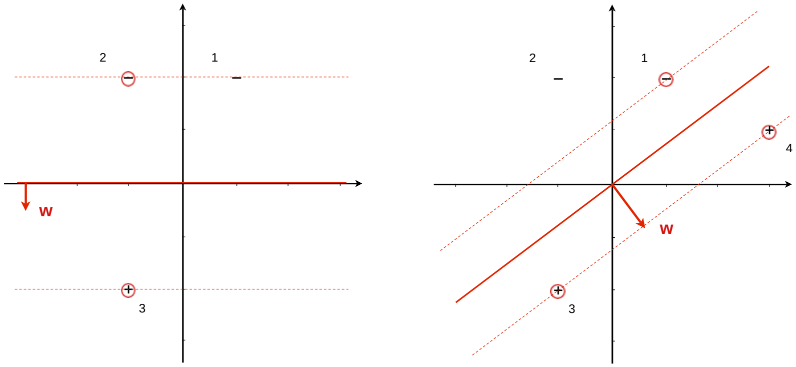
The dual optimisation problem for support vector machines is to maximise the dual Lagrangian under positivity constraints and one equality constraint:

$$\alpha_1^*,\dots,\alpha_n^* = \mathop{\arg\max}_{\alpha_1,\dots,\alpha_n} -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \mathbf{x}_i\cdot\mathbf{x}_j + \sum_{i=1}^{n}\alpha_i$$

$$\text{subject to } \alpha_i \geq 0, 1 \leq i \leq n \text{ and } \sum_{i=1}^{n}\alpha_i y = 0$$

Figure 7.8, p.215

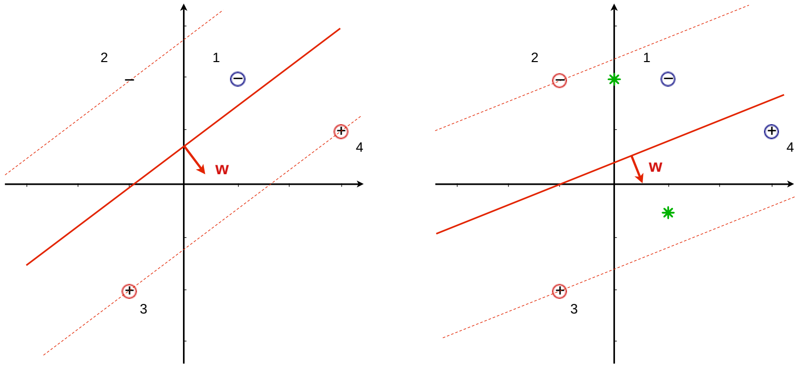
Two maximum-margin classifiers



(left) A maximum-margin classifier built from three examples, with $\mathbf{w} = (0, -1/2)$ and margin 2. The circled examples are the support vectors: they receive non-zero Lagrange multipliers and define the decision boundary. **(right)** By adding a second positive the decision boundary is rotated to $\mathbf{w} = (3/5, -4/5)$ and the margin decreases to 1.

Figure 7.9, p.218

Soft margins

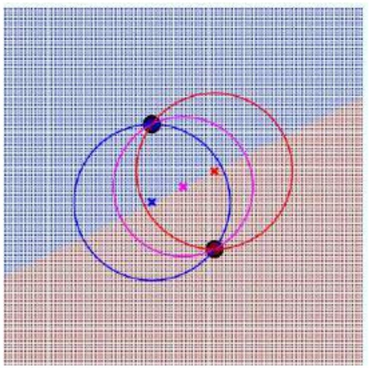


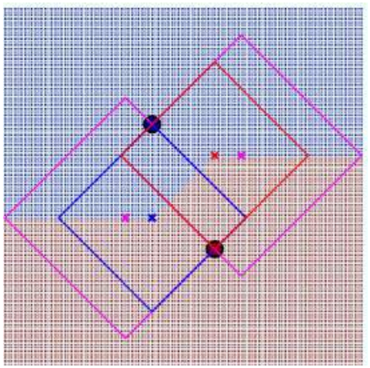
(left) The soft margin classifier learned with $C = 5/16$, at which point $\mathbf{x}_2$ is about to become a support vector. **(right)** The soft margin classifier learned with $C = 1/10$: all examples contribute equally to the weight vector. The asterisks denote the class means, and the decision boundary is parallel to the one learned by the basic linear classifier.

What's next?

Figure 8.6, p.240

Two-exemplar decision boundaries

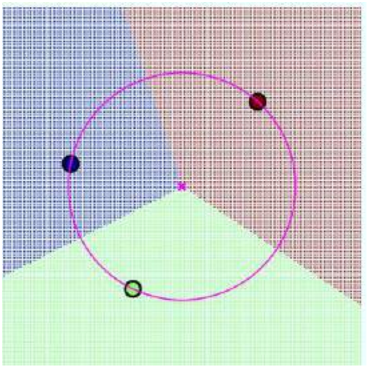


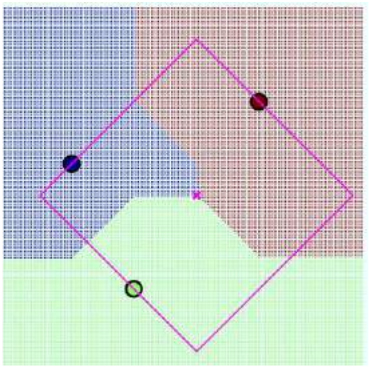


(left) For two exemplars the nearest-exemplar decision rule with Euclidean distance results in a linear decision boundary coinciding with the perpendicular bisector of the line connecting the two exemplars. **(right)** Using Manhattan distance the circles are replaced by diamonds.

Figure 8.7, p.240

Three-exemplar decision boundaries

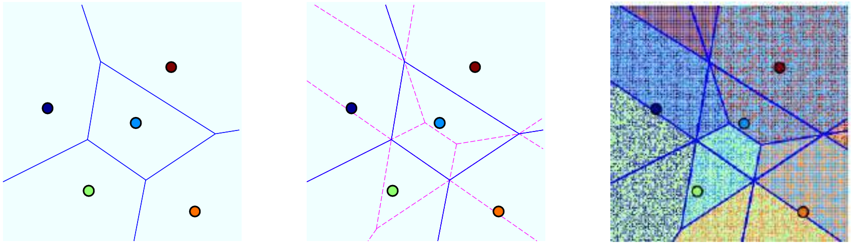




(left) Decision regions defined by the 2-norm nearest-exemplar decision rule for three exemplars. **(right)** With Manhattan distance the decision regions become non-convex.

Figure 8.8, p.241

One vs two nearest neighbours



(left) Voronoi tesselation for five exemplars. **(middle)** Taking the two nearest exemplars into account leads to a further subdivision of each Voronoi cell. **(right)** The shading indicates which exemplars contribute to which cell.

Distance-based models

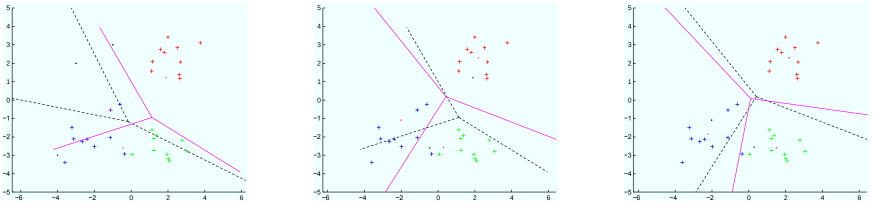
To summarise, the main ingredients of distance-based models are

- distance metrics, which can be Euclidean, Manhattan, Minkowski or Mahalanobis, among many others;
- exemplars: centroids that find a centre of mass according to a chosen distance metric, or medoids that find the most centrally located data point; and
- distance-based decision rules, which take a vote among the k nearest exemplars.

In the next subsections these ingredients are combined in various ways to obtain supervised and unsupervised learning algorithms.

Figure 8.11, p.248

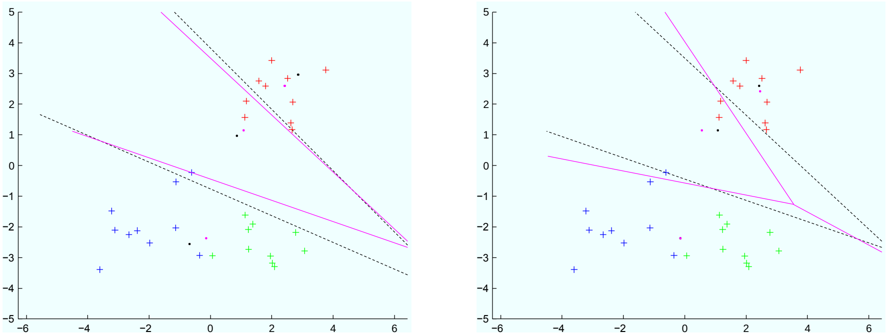
K -means clustering



(left) First iteration of 3-means on Gaussian mixture data. The dotted lines are the Voronoi boundaries resulting from randomly initialised centroids; the violet solid lines are the result of the recalculated means. **(middle)** Second iteration, taking the previous partition as starting point (dotted line). **(right)** Third iteration with stable clustering.

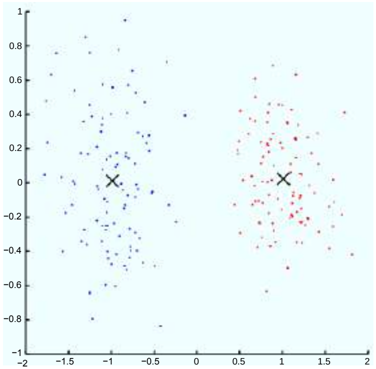
Figure 8.12, p.249

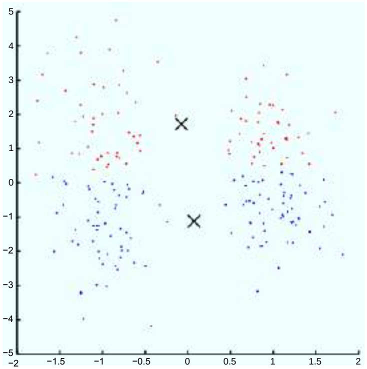
Sub-optimality of K-means



(left) First iteration of 3-means on the same data asFigure 8.11with differently initialised centroids. **(right)** 3-means has converged to a sub-optimal clustering.

Scale-sensitivity of K-means

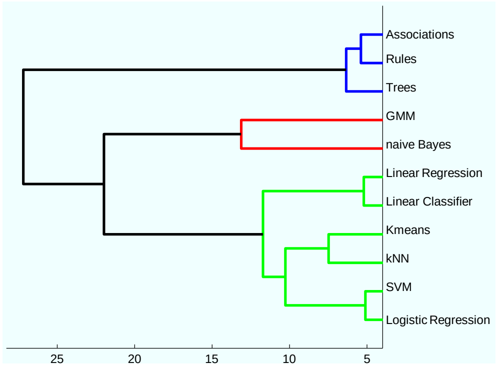




(left) On this data 2-means detects the right clusters. **(right)** After rescaling the y-axis, this configuration has a higher between-cluster scatter than the intended one.

Figure 8.15, p.253

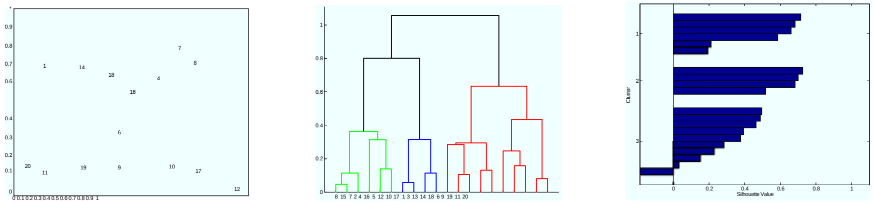
Hierarchical clustering example



A dendrogram (printed left to right to improve readability) constructed by hierarchical clustering from the data inTable 1.4.

Figure 8.18, p.258

A spurious clustering



(left) 20 data points, generated by uniform random sampling. **(middle)** The dendrogram generated from complete linkage. The three clusters suggested by the dendrogram are spurious as they cannot be observed in the data. **(right)** The rapidly decreasing silhouette values in each cluster confirm the absence of a strong cluster structure. Point 18 has a negative silhouette value as it is on average closer to thegreen pointsthan to the otherred points.

What’s next?

Discriminative and generative probabilistic models

t *Discriminative models* model the posterior probability distribution $P(Y|X)$, where Y is the target variable and X are the features. That is, given X they return a probability distribution over Y.

t *Generative models* model the joint distribution $P(Y, X)$ of the target Y and the feature vector X. Once we have access to this joint distribution we can derive any conditional or marginal distribution involving the same variables. $P(Y|X) = \frac{P(Y,X)}{\sum_y P(Y=y,X)}$.

t Alternatively, generative models can be described by the likelihood function $P(X|Y)$, since $P(Y,X) = P(X|Y)P(Y)$ and the target or prior distribution (usually abbreviated to 'prior') can be easily estimated or postulated.

Categorical random variables

Categorical variables or features (also called discrete or nominal) are ubiquitous in machine learning.

- Perhaps the most common form of the Bernoulli distribution models whether or not a word occurs in a document. That is, for the i-th word in our vocabulary we have a random variable X_i governed by a Bernoulli distribution. The joint distribution over the *bit vector* $X = (X_1, \dots, X_k)$ is called a *multivariate Bernoulli distribution*.
- Variables with more than two outcomes are also common: for example, every word position in an e-mail corresponds to a categorical variable with k outcomes, where k is the size of the vocabulary. The *multinomial distribution* manifests itself as a *count vector*: a histogram of the number of occurrences of all vocabulary words in a document.

Both these document models are in common use. Despite their differences, they both assume independence between word occurrences, generally referred to as the *naive Bayes assumption*.

Probabilistic decision rules

We have chosen one of the possible distributions to model our data X as coming from either class.

- The more different $P(X|Y=\text{spam})$ and $P(X|Y=\text{ham})$ are, the more useful the features X are for classification.
- Thus, for a specific e-mail x we calculate both $P(X=x|Y=\text{spam})$ and $P(X=x|Y=\text{ham})$, and apply one of several possible decision rules:

maximum likelihood (ML) – predict	$\arg\max_y P(X=x|Y=y)$;
maximum a posteriori (MAP) – predict	$\arg\max_y P(X=x|Y=y)P(Y=y)$;
recalibrated likelihood – predict	$\arg\max_y w_y P(X=x|Y=y)$.

The relation between the first two decision rules is that ML classification is equivalent to MAP classification with a uniform class distribution. The third decision rule generalises the first two in that it replaces the class distribution with a set of weights learned from the data.

Example 9.4, p.276

Prediction using a naive Bayes modelI

Suppose our vocabulary contains three words a, b and c, and we use a multivariate Bernoulli model for our e-mails, with parameters

$$\boldsymbol{\theta}^{\oplus} = (0.5, 0.67, 0.33) \qquad \boldsymbol{\theta}^{8} = (0.67, 0.33, 0.33)$$

This means, for example, that the presence of b is twice as likely in spam (+), compared with ham.

The e-mail to be classified contains words a and b but not c, and hence is described by the bit vector $\mathbf{x} = (1, 1, 0)$. We obtain likelihoods

$$P(\mathbf{x}|\oplus) = 0.5 \cdot 0.67 \cdot (1 - 0.33) = 0.222 \qquad P(\mathbf{x}|8) = 0.67 \cdot 0.33 \cdot (1 - 0.33) = 0.148$$

The ML classification of **x** is thus spam.

Example 9.4, p.276

Prediction using a naive Bayes modelII

In the case of two classes it is often convenient to work with likelihood ratios and odds.

- t The likelihood ratio can be calculated as

$$\frac{P(\mathbf{x}|\oplus)}{P(\mathbf{x}|8)} = \frac{0.5\ 0.67\ 1-0.33}{0.67\ 0.33\ 1-0.33} = 3/2 > 1$$

- t This means that the MAP classification of **x** is also spam if the prior odds are more than $2/3$, but ham if they are less than that.
- t For example, with 33% spam and 67% ham the prior odds are $\frac{P(\oplus)}{P(8)} = \frac{0.33}{0.67} = 1/2$, resulting in a posterior odds of

$$\frac{P(\oplus|\mathbf{x})}{P(8|\mathbf{x})} = \frac{P(\mathbf{x}|\oplus)\,P(\oplus)}{P(\mathbf{x}|8)\,P(8)} = 3/2 \cdot 1/2 = 3/4 < 1$$

In this case the likelihood ratio for **x** is not strong enough to push the decision away from the prior.

Table 9.1, p.280

Training data for naive Bayes

E-mail	#a	#b	#c	Class
e_1	0	3	0	+
e_2	0	3	3	+
e_3	3	0	0	+
e_4	2	3	0	+
e_5	4	3	0	–
e_6	4	0	3	–
e_7	3	0	0	–
e_8	0	0	0	–

E-mail	a?	b?	c?	Class
e_1	0	1	0	+
e_2	0	1	1	+
e_3	1	0	0	+
e_4	1	1	0	+
e_5	1	1	0	–
e_6	1	0	1	–
e_7	1	0	0	–
e_8	0	0	0	–

(left) A small e-mail data set described by count vectors. **(right)** The same data set described by bit vectors.

Example 9.5, p.279

Training a naive Bayes modelI

Consider the following e-mails consisting of five words a, b, c, d, e:

e_1: $b\ d\ e\ b\ b\ d\ e$	e_5: $a\ b\ a\ b\ a\ b\ a\ e\ d$
e_2: $b\ c\ e\ b\ b\ d\ d\ e\ c\ c$	e_6: $a\ c\ a\ c\ a\ c\ a\ e\ d$
e_3: $a\ d\ a\ d\ e\ a\ e\ e$	e_7: $e\ a\ e\ d\ a\ e\ a$
e_4: $b\ a\ d\ b\ e\ d\ a\ b$	e_8: $d\ e\ d\ e\ d$

We are told that the e-mails on the left are spam and those on the right are ham, and so we use them as a small training set to train our Bayesian classifier.

- First, we decide that d and e are so-called *stop words* that are too common to convey class information.
- The remaining words, a, b and c, constitute our vocabulary.

Example 9.5, p.279

Training a naive Bayes modelII

In the multivariate Bernoulli model e-mails are represented by bit vectors, as in Table 9.1 (right).

- Adding the bit vectors for each class results in $(2,3,1)$ for spam and $(3,1,1)$ for ham.
- Each count is to be divided by the number of documents in a class, in order to get an estimate of the probability of a document containing a particular vocabulary word.
- Probability smoothing now means adding two pseudo-documents, one containing each word and one containing none of them.
- This results in the estimated parameter vectors $\hat{\boldsymbol{\theta}}^{\oplus} = (3/6,4/6,2/6) = (0.5,0.67,0.33)$ for spam and $\hat{\boldsymbol{\theta}}^{8} = (4/6,2/6,2/6) = (0.67,0.33,0.33)$ for ham.

What’s next?

Feature statistics

Three main categories are *statistics of central tendency*, *statistics of dispersion* and *shape statistics*. Each of these can be interpreted either as a theoretical property of an unknown population or a concrete property of a given sample – here we will concentrate on sample statistics.

Starting with statistics of central tendency, the most important ones are

- the *mean* or average value;
- the *median*, which is the middle value if we order the instances from lowest to highest feature value; and
- the *mode*, which is the majority value or values.

Categorical, ordinal and quantitative features

Given these various statistics we can distinguish three main kinds of feature: those with a meaningful numerical scale, those without a scale but with an ordering, and those without either.

- We will call features of the first type *quantitative*; they most often involve a mapping into the reals (another term in common use is ‘continuous’).
- Features with an ordering but without scale are called *ordinal features*. The domain of an ordinal feature is some totally ordered set, such as the set of characters or strings. Another common example are features that express a rank order: first, second, third, and so on. Ordinal features allow the mode and median as central tendency statistics, and quantiles as dispersion statistics.
- Features without ordering or scale are called *categorical features* (or sometimes ‘nominal’ features). They do not allow any statistical summary except the mode. One subspecies of the categorical features is the *Boolean feature*, which maps into the truth values true and false.

Table 10.1, p.304

Kinds of feature

Kind	*Order*	*Scale*	*Tendency*	*Dispersion*	*Shape*
Categorical	×	×	mode	n/a	n/a
Ordinal	✓	×	median	quantiles	n/a
Quantitative	✓	✓	mean	range, interquartile range, variance, standard deviation	skewness, kurtosis

Kinds of feature, their properties and allowable statistics. Each kind inherits the statistics from the kinds above it in the table. For instance, the mode is a statistic of central tendency that can be computed for any kind of feature.

Table 10.2, p.307

Feature transformations

↓ to, from →	*Quantitative*	*Ordinal*	*Categorical*	*Boolean*
Quantitative	**normalisation**	**calibration**	**calibration**	**calibration**
Ordinal	**discretisation**	**ordering**	**ordering**	**ordering**
Categorical	**discretisation**	**unordering**	**grouping**	
Boolean	**thresholding**	**thresholding**	**binarisation**	

An overview of possible feature transformations. **Normalisation and calibration** adapt the scale of quantitative features, or add a scale to features that don't have one. **Ordering** adds or adapts the order of feature values without reference to a scale. The other operations abstract away from unnecessary detail, either in a deductive way (**unordering**, **binarisation**) or by introducing new information (**thresholding**, **discretisation**).

What’s next?

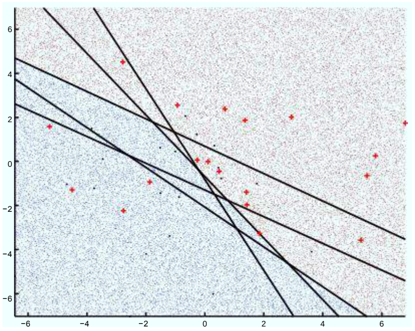
Ensemble methods

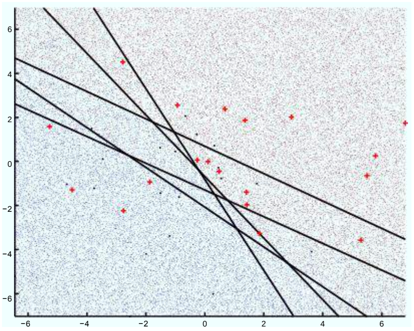
In essence, ensemble methods in machine learning have the following two things in common:

- they construct multiple, diverse predictive models from adapted versions of the training data (most often reweighted or resampled);
- they combine the predictions of these models in some way, often by simple averaging or voting (possibly weighted).

Figure 11.1, p.332

Bagging





(left) An ensemble of five basic linear classifiers built from bootstrap samples with bagging. The decision rule is majority vote, leading to a piecewise linear decision boundary. **(right)** If we turn the votes into probabilities, we see the ensemble is effectively a grouping model: each instance space segment obtains a slightly different probability.

Algorithm 11.2, p.333

Random forests

Algorithm RandomForest(D, T, d)– train an ensemble of tree models from bootstrap samples and random subspaces.

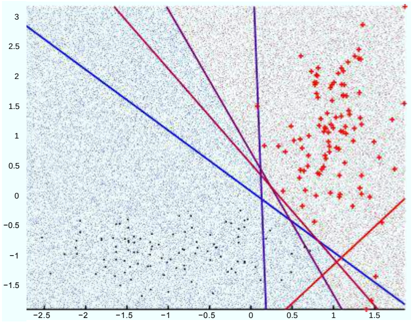
Input : data set D; ensemble size T; subspace dimension d.

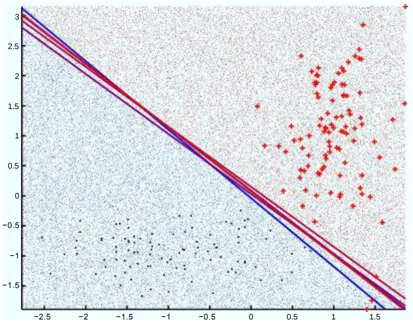
Output : ensemble of tree models whose predictions are to be combined by voting or averaging.

```
1 for $t = 1$ to $T$ do
2     build a bootstrap sample $D_t$ from $D$ by sampling $|D|$ data points with replacement;
3     select $d$ features at random and reduce dimensionality of $D_t$ accordingly;
4     train a tree model $M_t$ on $D_t$ without pruning;
5 end
6 return $\{M_t | 1 \le t \le T\}$
```

Figure 11.2, p.336

Boosting





(left) An ensemble of five boosted basic linear classifiers with majority vote. The linear classifiers were learned fromblue tored; none of them achieves zero training error, but the ensemble does. **(right)** Applying bagging results in a much more homogeneous ensemble, indicating that there is little diversity in the bootstrap samples.

What’s next?

Machine learning experiments

Machine learning experiments pose questions about models that we try to answer by means of measurements on data.
The following are common examples of the types of question we are interested in:

- How does model m perform on data from domain D?
- Which of these models has the best performance on data from domain D?
- How do models produced by learning algorithm A perform on data from domain D?
- Which of these learning algorithms gives the best model on data from domain D?

Assuming we have access to data from domain D, we perform measurements on our models using this data in order to answer these questions.

What to measure

Your choice of evaluation measures should reflect the assumptions you are making about your experimental objective as well as possible contexts in which your models operate. We have looked at the following cases:

- Accuracy is a good evaluation measure if the class distribution in your test set is representative for the operating context.
- Average recall is the evaluation measure of choice if all class distributions are equally likely.
- Precision and recall shift the focus from classification accuracy to a performance analysis ignoring the true negatives.
- Predicted positive rate and AUC are relevant measures in a ranking context.

Example 12.4, p.350

Cross-validation

The following table gives a possible result of evaluating three learning algorithms on a data set with 10-fold cross-validation:

Fold	*Naive Bayes*	*Decision tree*	*Nearest neighbour*
1	0.6809	0.7524	0.7164
2	0.7017	0.8964	0.8883
3	0.7012	0.6803	0.8410
4	0.6913	0.9102	0.6825
5	0.6333	0.7758	0.7599
6	0.6415	0.8154	0.8479
7	0.7216	0.6224	0.7012
8	0.7214	0.7585	0.4959
9	0.6578	0.9380	0.9279
10	0.7865	0.7524	0.7455
avg	0.6937	0.7902	0.7606
stdev	0.0448	0.1014	0.1248

The last two lines give the average and standard deviation over all ten folds. Clearly the decision tree achieves the best result, but should we completely discard nearest neighbour?

Current topics in machine learning

t Deep learning, big data, structured data

t Ethics, privacy and trustworthiness

t Fairness, accountability and transparency

t Multi-label and preference learning; structured outputs

t Multi-task and transfer learning

t Online learning, data streams, active learning

t Reinforcement learning