

## Chapter 2

# Feature Selection and Regression

### 2.1 Introduction

In food design, models explain the relationship between the sensory features and consumer preferences of a product. Model building is an essential part of this data analysis. Feature selection is an essential part of model building: by reducing the number of features, we also reduce the number of model parameters. This makes estimating the parameters easier, and hence allows more accurate models to be built. Being smaller, the models may also gain clarity. In the absence of sufficient domain knowledge, the data itself must be used for feature selection. However, very small data sets contain very little information, making feature selection problematic.

Finding a useful set of features is a form of search, where the space we are searching grows factorially with the number of features, so that even an apparently modest number of features creates an intractable problem. This forces us to use heuristic search methods. The problems are exacerbated by the lack of data, giving us very little information for evaluating decisions.

In this chapter, we describe several widely used feature selection techniques from the fields of statistics and machine learning. However, these are shown to overfit the data, i.e. to select too many features and therefore build too complex a model. This motivates a discussion of several methods which limit the complexity of the models, and hence the number of features. It is argued that many of these techniques are inappropriate when dealing with very small data sets. This leads us to the conclusion that cross model validation (which is used to limit the number of features selected — see Section 2.5.1), combined with heuristic feature selection, provides the best solution to feature selection problems.

It is argued that a better approach would be to create and analyse data sets with far fewer features than is currently the norm. This would require a change in the way data is gathered, and would allow exhaustive search techniques to be used, which would

guarantee finding the optimum set of features with respect to the data, something which is currently impossible.

This chapter interleaves work on regression and on feature selection, rather than treating them as two separate fields. Kohavi and John [75] argue that the two should be combined because

the optimal features depend on the specific biases and heuristics of the learning algorithm [75, Section 10].

Therefore, in this work, feature sets are regarded as good or bad only with respect to the particular regression model being used.

In some food studies, the sensory panel is presented with more products than the preference panel. This has the effect of generating some extra unlabelled data (i.e. records with no known preference scores) at no extra cost. This data can be used to generate improved models, both for feature selection and for regression. In this chapter, we present two new algorithms: a semi-supervised feature selection algorithm (Section 2.8), and a semi-supervised ensemble regression algorithm (Section 2.11).

In Section 2.2, we describe the two regression algorithms used in this work, and discuss whether or not consumer preference scores are linear with respect to sensory attribute scores. In Sections 2.3–2.4, we discuss several approaches to feature selection. In Section 2.5, we discuss approaches to avoiding overfitting, especially the dangers of selecting too many features from a small data set. In Section 2.7, we describe the results of several feature selection methods applied to the food data sets. In Section 2.8, we describe a novel approach to feature selection, using semi-supervised learning. In the remainder of the chapter, we discuss other aspects of regression: using the Minimum Description Length to avoid overfitting (Section 2.9); using the EM algorithm (Section 2.10); a novel method for building semi-supervised ensembles (Section 2.11); and the affect on feature selection of the initial number of features (Section 2.12).

### 2.1.1 Selecting sensory features

As we described earlier (Section 1.4, page 24), when a sensory panel is formed, its members are charged with the task of defining a set of attributes of the food under examination, such that a) the panellists can measure reliably each product on each scale, and b) the attributes are sufficient to distinguish reliably between all the sample products. The tendency is to measure everything that can be measured, and therefore to produce a set of attributes that is larger than strictly necessary.

In sharp contrast, the consumers forming a preference panel are only presented with a very limited number of products, typically between 12 and 20. This is due to taste exhaustion: if a consumer tastes too many products over a limited time-period,

the responses will become close to random. Note that the sensory panel may formally measure only a similar small range of products, but will usually have analysed many more during the sensory panel's training period. The data we use in this work, and described in Section 1.4.4, consists only of measurements of the final product sets.

One of the main tasks in analysing food data is to build models relating sensory scores to preference scores, and it is on this that we now focus. When building models, we need more records than there are free parameters; otherwise an infinite series of models will fit the data equally well (such as drawing a regression line through just one point). Note that having more records than free parameters does not guarantee that we can find a useful model: the records may be correlated, and so not provide sufficient distinct information to guide the model building process. Given the relative sizes of the sensory and preference data sets, we want to reduce the number of features.

Rather than build a model that only fits the small data sample we have, we want to build models that fit the underlying distribution. This will allow us to make predictions about novel products, and so optimise their design. As with any science, there is inevitably a degree of noise in all of the measurements. A model which is very complex may fit the given data sample very well, but will fail to generalize well. I.e. the model will tend to make very poor predictions on future, unseen examples.

This is particularly problematic when modelling small data sets, owing to the shortage of information about the underlying distribution. Therefore we must be careful to avoid overfitting the data. If we use models of a single class (e.g. linear regression models) then the complexity depends only on the number of features. We must therefore determine the number of features to use as well as selecting the features themselves.

In consumer science, a *driver* is a product feature that strongly influences consumers' preferences, either positively or negatively. The identification of drivers is an end in itself when analysing consumer preference data: if we can identify the few features that are particularly important, then it is much easier to design the "perfect" product that matches the consumers' preferences.

If we were working with large data sets (i.e. with many products), then a further motivation behind feature selection would be speed. Using millions of records to estimate parameters, or using a model to make millions of predictions, takes a long time, and gets slower as we increase the number of features. With very small data sets however, we have the opposite effect: computationally intensive methods may become feasible owing to the small number of records.

In summary, we must select features to:

- understand consumers preferences;
- avoid overfitting; and

- allow model building to proceed with a small data set.

### 2.1.2 Relevant features

Most of the literature on feature selection concentrates on identifying and selecting all of the relevant features, while discarding all of the irrelevant ones (e.g. [2, 75, 70]). While appealing, this may be inappropriate when working with very small data sets, as discussed in Section 2.5 below.

John et al. [70] define three aspects of feature relevancy. All three definitions are based on considering the removal of a feature from a current set of features. A feature is *strongly relevant* if removing it from the current set necessarily leads to a worse performance of the regression or classification model. A feature is *weakly relevant* if removing it from the current set sometimes leads to a worse performance, but not always (depending on the other members of the feature set). A feature is *irrelevant* if removing it never degrades performance. Equivalent descriptions can be given in terms of adding features and improving performance. The overall aim of feature selection is usually defined as selecting all the strongly relevant features; as many weakly relevant features as are necessary; and no irrelevant features. Later work by Kohavi and John [75] shows that the optimum feature set<sup>1</sup> is not necessarily identical to the set of relevant features. In fact, relevance does not imply optimality, and optimality does not imply relevance [75, Section 2.3]. Certainly, selecting only relevant features is unlikely to be optimal when given a small data set: there may be insufficient evidence to support choosing *all* the strongly relevant features, let alone the weakly relevant ones. Again, we must be careful to control the number of features selected.

It is worth stressing that a feature set can only be considered “ideal” with respect to a particular problem. If we want to show that apples and oranges are similar, we might use the “shape” feature, as they are both spherical. If we want to highlight their *difference*, we might measure their colour. The Ugly Ducking theorem [39, p.458–461] shows that the quality of a set of features is always problem-dependent. Our problem is to predict product preferences from a set of sensory features, so we now turn to consider predictive models, and regression in particular.

## 2.2 Regression

Our aim here is to build predictive models of consumer preferences. Two options present themselves: classification and regression. We could in principle split food preferences into two opposing classes, such as “like” and “dislike”. However, this division is too crude to guide food design, because it fails to capture consumers’ responses to

---

<sup>1</sup>Optimal with respect to the estimated prediction accuracy of the final model.

food. Susmaga [122] describes using rough sets to form multi-class discretizations of continuous data, and compares this with other methods of discretization based on the number of partitions, the fraction of these that are redundant, and so on. Susmaga argues in favour of discretization by pointing out that discrete data is often faster to analyse, allows smaller models to be built, and is required by some algorithms (such as ID3, a common decision tree algorithm [100]). These advantages don't apply to very small data set analysis: the models will be fast to build, they will be small, and we can easily choose suitable algorithms that work with continuous data.

Torgo and Gama [129] describe a generic pre-processing step, to convert regression problems into classification problems. They combine several discretization methods with several classification algorithms, and compare the results to several regression methods, using a number of “real-world” data sets. Overall, they show that regression tends to produce more accurate results than discretized classification methods. Susmaga [122] does not provide any such comparison to regression methods. Even though classification is studied more widely in the machine learning community than is regression [129], there is no guarantee that classification models are more *accurate* than regression models, shown both by Torgo and Gama's results, and more generally by the No Free Lunch theorems (e.g. [135]).

The overheads of choosing a suitable method of discretization and a suitable cost function, coupled with the sparseness that such methods would produce from a small data set, strongly suggest that regression of continuous data is more appropriate. This decision also follows Marr's “Principle of Least Commitment” [77], which states that we should retain as much information as possible in our data, during all stages of processing. Given that our original (continuous) data is noisy, any attempt at discretization may lead to some values being assigned to the “wrong” bin, which would be an irrecoverable mistake. We therefore choose to use regression models rather than classification models in the remainder of this work. We now discuss two regression algorithms that we will use in later experiments, and consider why feature selection is vital for each of them.

### 2.2.1 Linear regression

A linear regression model defines a straight line passing through the data sample. The output is a simple weighted sum of the inputs, with an extra bias term. Given  $m$  variables and  $n$  records, this becomes:  $\mathbf{y} = \mathbf{A}\mathbf{u}$ , where  $\mathbf{y}$  is a vector of target values  $\mathbf{A}$  is an  $n \times (m+1)$  matrix of the input data, with an extra column of ones corresponding with the bias term; and  $\mathbf{u}$  is the vector of parameters we need to estimate, with  $m$  elements defining the gradient of the line and one element defining the vertical offset (bias). There are thus  $m + 1$  free parameters for a model of a data set containing  $m$  independent

variables. The optimum value of these free parameters is usually defined as the least squares solution, i.e. the parameters that minimise the error  $\mathbf{e}^2 = (\mathbf{y} - \mathbf{A}\mathbf{u})^2$ . This least squares problem can be solved through matrix algebra, by estimating  $\bar{\mathbf{u}} = (\mathbf{A}^t\mathbf{A})^{-1}\mathbf{A}^t\mathbf{y}$ , where  $\bar{\mathbf{u}}$  is the vector that minimises the errors  $\mathbf{e}$  [58]. This is straightforward and fast to calculate, assuming that  $\mathbf{A}$  is small enough to allow easy inversion. If there are fewer records than features, then calculating the weights  $\bar{\mathbf{u}}$  becomes impossible:  $\mathbf{A}^t\mathbf{A}$  becomes singular<sup>2</sup>, which makes the inversion numerically unstable. Thus, we may have to perform feature selection *before* building a linear regression model.

Once a regression model has been built, the results are easy to interpret: a large element of  $\bar{\mathbf{u}}$  corresponds to a variable that is an important driver (assuming that the data matrix  $\mathbf{A}$  has normalised rows and columns). If the weight is negative, the consumer dislikes that feature; if the weight is positive, the consumer likes that feature; if the weight is close to zero, that feature is irrelevant. Further analysis is also possible, such as using singular value decomposition of  $\mathbf{A}$ , and examining any dominant values.

If the underlying distribution from which the training data is sampled is not approximately linear in the area of interest, then the linear regression model will fail to capture this non-linearity, and so perform badly.

## 2.2.2 Radial basis function networks

Radial basis function (RBF) networks are one of the most common varieties of feed-forward networks, although their original purpose was to perform exact interpolation of a set of data [98]. RBF networks use a linear combination of non-linear local kernel functions, which allows them to find good approximations to complex functions. A typical RBF network consists of a number of Gaussian basis functions, which map the input data into some other space, with a bias term to allow non-zero offset. The output is the weighted sum of the Gaussian activations and the bias. The output  $y$  of an RBF network with  $h$  basis functions  $\phi_i$ , and weight terms  $w_i$ , given input vector  $\mathbf{x}$  is:

$$y(\mathbf{x}) = \sum_{i=1}^h w_i \phi_i(\mathbf{x}) + w_0.$$

The kernel (or basis) function  $\phi$  used in this work is exponential:

$$\phi_i(\mathbf{x}) = \exp\left(-\frac{\|\mathbf{x} - \boldsymbol{\mu}_i\|^2}{2\sigma_i^2}\right)$$

where  $\boldsymbol{\mu}_i$  is the centre of the  $i^{th}$  basis function, which has variance (width)  $\sigma_i^2$ .

---

<sup>2</sup>A singular matrix is a square matrix whose determinant is equal to zero, and which therefore cannot be inverted.

The training algorithm used in this work has two stages: first, the expectation-maximisation (EM) algorithm<sup>3</sup> is used to estimate the means and variances of the Gaussian components; then the weight and bias parameters are calculated by matrix inversion, similar to that used in linear regression, described above. This two stage approach is described by Bishop [15, p.170]. Other forms of RBF networks use different basis functions, or introduce further constraints, such as having a fixed width for all the basis functions. There are many other training algorithms used for estimating the parameters; we use the two-stage process described above principally for reasons of speed (e.g. [15, p.183]). Recent authors, such as Schölkopf et al. [113], treat RBF networks as a special case of support vector machines (SVMs). SVMs use kernels to project data into a different space, where classification (or regression) becomes easier. Using Gaussian kernel produces a system equivalent to an RBF network, but using other kernels (such as polynomials) gives SVMs greater flexibility. However, SVMs have extra parameters to be selected (such as the choice of the kernel function, and the value of ‘C’), are typically slower to use than RBF networks, and have no guarantee of being more accurate. We will not consider them further.

Unlike linear regression, RBF networks can approximate any function to an arbitrarily small error, given enough basis functions, and therefore enough data to estimate the corresponding parameters (Hartman et al. [63]). Unlike linear regression, we can build RBF networks even if we have fewer records than features. The second stage of building the network only requires that we have fewer Gaussian components than records to allow matrix inversion, and not necessarily fewer features. However, the estimation of the parameters is likely to be poor, owing to the lack of information, so feature selection is still essential.

## 2.3 Approaches to feature selection

One traditional approach to feature selection is to use correlation analysis to estimate whether a feature is likely to be useful (e.g. a sensory feature correlated with the preference) or redundant (e.g. two sensory features that are strongly correlated). This technique is an example of a filter: we take a set of data, filter out the unwanted features, and then build a model from the remainder (see Figure 2.1).

The main alternative to feature selection is feature extraction, where new features are created by transforming the original set, typically incorporating dimension reduction. In both cases, we reduce the number of parameters required by the model, but feature extraction achieves this by transforming the data before simplifying it. One of the most widely used techniques is principal components analysis, or PCA (e.g.

---

<sup>3</sup>The EM algorithm is described further in Section 3.4.5. The appendix gives further implementation details in Section A.1.1.

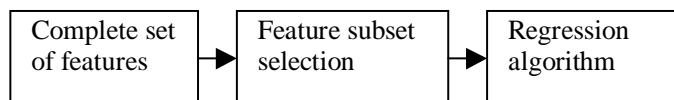


Figure 2.1: Filter methods for feature selection

Gnanadesikan [52]). This creates new features by forming orthogonal, uncorrelated linear combinations of the original features, and then selecting a few of the most important derived features. The aim is to find a low-dimensional representation of the data that accounts for most of the variance in the original features. Factor analysis [52] is a closely related method which aims to account for the correlations (rather than variance) between the original features.

However, in the case of food design, dimension reduction by itself is not our only goal: we also want to understand *why* consumers have their preferences, described in terms of food properties i.e. using the same measures that the sensory panel derived. This suggests that feature extraction by itself is not useful, unless it also allows us to identify and select features from the sensory panel. Another potential problem with PCA and related methods, is that by selecting a subset of the derived components, we may be throwing away information that is discriminatory, even if only weakly so. This goes against Marr’s “Principle of Least Commitment” [77] mentioned earlier (Section 2.2).

Duda and Hart give a simple hierarchical dimension reduction algorithm [38, p.247], which works by iteratively merging pairs of features that are most highly correlated. As we want to retain the original features to aid interpretability, this merging is unacceptable, but we can adapt Duda and Hart’s algorithm to perform feature selection rather than feature extraction. Figure 2.2 shows a form of backwards elimination, where the two features which are most highly correlated with each other are selected, and whichever one is least correlated with the target is removed. This process continues until no features are left. At each stage, a model can be built and its generalization error estimated (e.g. using cross validation, as described in Section 2.5.1). We can then terminate the algorithm when the error starts rising. The results presented in Section 2.7 include this “correlation based feature elimination” (CBFE) technique.

One alternative to filtering out unwanted features is embedded feature selection, which describes induction algorithms that perform feature selection as a side effect. For example, when building a decision tree, the most informative variable is selected as the root, with less informative variables appearing lower down the tree or being entirely absent. Breiman et al. [20, p.217ff] discuss the Boston housing data (which we analyse in Section 2.11 below), and build a regression tree. When predicting house prices, the first variable selected by the algorithm is the number of rooms. Other key indicators

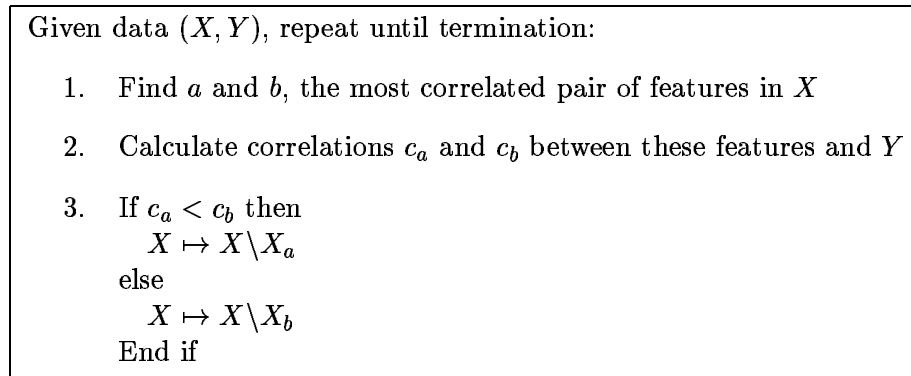


Figure 2.2: Correlation based feature elimination

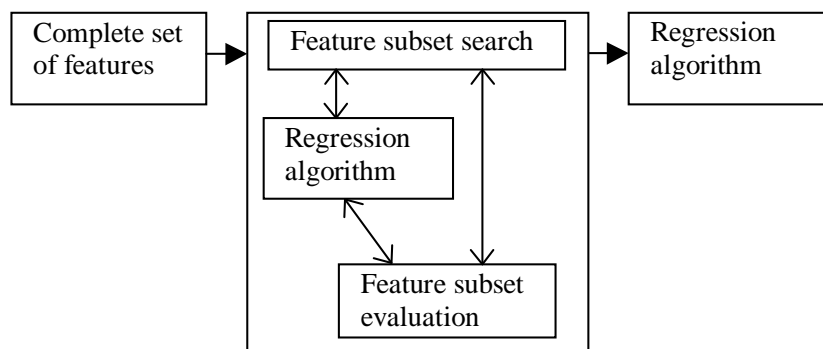


Figure 2.3: Wrapper methods for feature selection

appearing near the top of the tree include crime rate and the percent of “lower-status population” in each area; the original variable of interest (air pollution) was not used at all. Thus, certain regression algorithms intrinsically perform feature selection.

Unlike filters and embedded methods, wrapper approaches use the regression model (or some other inductive algorithm) to evaluate many possible sets of features, and then choose one set that generalizes well to build the final model (Figure 2.3). We wrap a feature selection algorithm around the regression model, with the aim of selecting features that minimise the generalization error. In order to use a wrapper, we must define a space of models, and a method for searching it. This is the subject of Section 2.4. Note that filter methods can still be thought of as searching through the space of feature subsets, albeit a deterministic search that only covers a very small part of the space. With wrappers, there is greater freedom in exactly how the model space is defined, and how it is then searched.

We want to find a set of features that will allow us to build a good model. Wrapper methods explicitly define a good set of features as a set which produces an accurate predictive model, rather than using a more abstract definition based on (for example) correlation or variance, which may or may not produce a useful regression model.

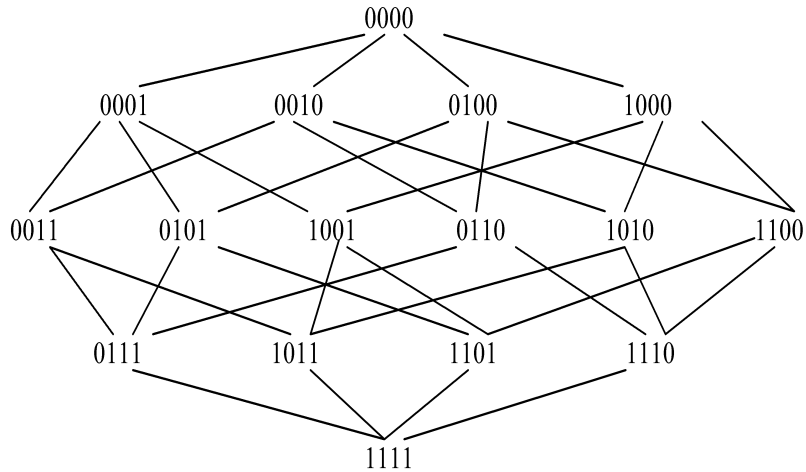


Figure 2.4: Space of feature subsets (based on Kohavi and John [75]). Not all of the possible paths through the model space are explicitly shown.

## 2.4 Searching with wrappers

We now focus on wrapper approaches to feature selection, discussing the feature space, its size, and how to search it effectively. Figure 2.4 shows one representation of a feature subset space, with a total of four features. At the top, we have the single model that contains no features at all (represented by four zeros), and at the bottom the single model containing all features (represented by four ones). The lines linking pairs of models represent the addition (moving down the graph) or removal (moving up) of a single feature. There are many ways to explore this space, not all of which directly follow these lines.

Two general problems with the search for feature subsets should be highlighted. Firstly, in common with many search problems, the value of the optimal solution is unknown: we don't know in advance the regression error associated with the best possible feature set. We are looking for a needle in a haystack — and we don't know what needles look like. Secondly, unlike many search problems, feature selection provides no gradient information. Each feature is either used in the model or not, creating a search landscape consisting of horizontal terraces and vertical cliffs. This means that the direction of the search is random; we have to take a step, evaluate the results, and then either move on or move back.

### 2.4.1 Exhaustive search

As the name suggests, we could “simply” try out every possible combination of features. However, there are  $2^N$  possible subsets drawn from  $N$  features; with the 65 features of the meat sensory data set, this gives  $\sim 4 \times 10^{19}$  combinations, which would take

thousands of years to evaluate. Such exhaustive searching is computationally infeasible if the data set has more than about 20 – 25 features (and hence  $\sim 3 \times 10^7$  combinations)<sup>4</sup>. One option is to use some other technique to remove the least likely features (e.g. one of a pair of strongly correlated features), and then exhaustively search across the “best” 25 features (or fewer). However, this amounts to exhaustively searching just one small part of the original space, and unless we are certain that the optimum answer is within this subspace, it will ultimately be misleading.

This assumes that we do not know the error associate with the most accurate model. If such an optimal solution is known, then we can use efficient methods to search for it, such as “depth-first branch-and-bound” or the  $A^*$  algorithms [138]. When the optimum solution is unknown, as in our case, we are forced to use heuristic search methods. This is because if we do not know the minimum error, then we will not know when we have found the optimal model until we have evaluated every possible model. We need some approximation to allow an early termination of the search to make it feasible. Historically, early attempts at developing such heuristic methods lead to approximate versions of the previous exhaustive search methods, such as beam search, which is an approximate version of brach-and-bound [138, p. 25]. These approaches were designed for moderate-sized problem spaces, where they are very effective. However, even these approximations cease to be useful as the problem space grows. Rayward-Smith et al. [103] argue that these methods perform no better than exhaustive enumeration, in the sense that the time (or memory) complexity grows too rapidly to be useful. While it is possible that some form of beam search would improve on the results we present here, such alternatives are left to future work. In the remainder of this section, we discuss heuristic approaches to early stopping, such as sequential and stochastic algorithms.

### 2.4.2 Sequential methods

The most widely used sequential method is forward sequential selection, as described by Theodoridis and Koutroumbas [126, p.160]. Here, we build a feature subset by greedily adding one feature at a time to an initially empty set. To add feature  $n+1$  to an existing subset of  $n$  features from a total of  $N$  features, we build  $(N - n)$  models, each consisting of the current  $n$  features and one extra. We then select the single feature that reduces the model’s error the most, and add that to our current feature set. We continue to add features in this way until some pre-specified stopping condition is reached, such as when the addition of a new feature does not reduce the error of the model. In Figure 2.4, this can be shown as starting at the top, testing each available link in turn, and then following a single link down through each level, while never moving sideways or

---

<sup>4</sup>Suppose it takes one day for a computer to exhaustively search a space of 25 features; then adding one extra feature will take almost a month, and two extra will take almost two years.

upwards. The model error is typically defined as the generalization error, as estimated using a hold-out set (Section 2.5.1).

The complementary method is backwards elimination selection, which starts with a model using all the features, and greedily removes the single feature that lowers the generalization error by the greatest amount. This is equivalent to traversing Figure 2.4 from bottom to top. One drawback is that because we want more records than features, it may be impossible or undesirable to build the first model required (i.e. one containing all features).

One open problem with forward sequential selection is selecting a stopping criterion, i.e. a criterion for determining when to stop adding features. Given sufficient data, one can happily continue to add features until the error stops dropping, but with a very small data set this often leads to overfitting. A more formal statistical approach is based on observing the distribution of how the error changes when adding a single feature to the current set. At the start, we evaluate the “no feature” model<sup>5</sup>, and then build  $N$  models, each containing exactly one of the  $N$  features. Each of these models will have an error associated with it, which we can compare with the error of the “no feature” model. We can then produce a histogram showing how often the error decreases or increases as we add features. The best and worst features will be at the extremes. In order to justify including the best feature in the model, we need to verify not only that the error is reduced by its inclusion, but also that this reduction is significant. In the simplest case, we could fit a normal distribution to the error changes (taking the empirical mean and standard deviation from the histogram), and test to see if the best feature is in the 5% tail of the distribution. A  $t$ -test may be more robust, given that the histogram would be formed from a very small sample. In the results section below, we describe this as a “Monte Carlo significance test” (Section 2.7, p. 61).

All sequential techniques tend to suffer owing to their greediness. Once forward sequential selection has added a feature, that feature will remain in all subsequent models, and its removal will never be considered. One attempt to offset this is to combine forwards and backwards sequential techniques, into floating sequential selection (e.g. Theodoridis and Koutroumbas [126, p.161]). There are many forms of this, but the general idea is to perform forward selection, followed by backwards elimination, then forward again, and so on. This adds a modest and controllable amount of computational expense, and introduces further parameters to be estimated, but brings the advantage that it selects combinations of features that work well together, rather than selecting one feature at a time.

Floating methods by no means guarantee finding high quality solutions, as any sequential technique will tend to become stuck in local optima (e.g. [126, p.160]). At

---

<sup>5</sup>A model with no features will produce a constant output; e.g.  $y = 5.6$ .

some point in our search, we will find that any single change we can make (i.e. adding or removing a single feature) leads to a worse model. This does not mean, however, that we have found the optimum solution: it might just be that we need to change the model more substantially to improve it. Various attempts to avoid the solution becoming trapped in such local optima have inspired a number of heuristic techniques and other methods, such as simulated annealing, which we describe next.

### 2.4.3 Simulated annealing

Simulated annealing (SA) consists of a random walk through the search space, each step consisting of the addition or removal of a single feature (e.g. Rayward-Smith et al. [103]). It is inspired by annealing: when a piece of metal is heated and then allowed to cool, its atoms vibrate rapidly at first, but gradually slow down, settling into a regular lattice arrangement with (locally) minimum energy. At early stages of the simulation, the “temperature” is high, allowing steps to be taken at random, even if they make the model worse. As the temperature drops however, only steps that improve the model tend to be allowed. Thus, SA is a minimisation technique.

Potential steps (i.e. the addition or removal of a feature) are selected randomly and evaluated. If the step lowers the error, it is always accepted. Otherwise, the probability of taking a locally worse step<sup>6</sup> is directly proportional to the change in error, and inversely proportional to the temperature. The probability of moving at any one step from feature set  $s_i$  with error  $e_i$  to feature set  $s_j$  with error  $e_j$ , while the temperature is  $T$ , is given by:

$$p(s_i \rightarrow s_j) = \begin{cases} \exp(-(e_j - e_i)/T), & \text{if } e_j \geq e_i; \\ 1 & \text{otherwise.} \end{cases}$$

Thus, steps which make the model worse are taken occasionally, but mostly at high temperatures (i.e. early in the algorithm). This enables the algorithm to overcome local optima, but still allows a more delicate, localised search in the later stages. Unfortunately, the algorithm is still not guaranteed to find the global optimum, unless an unrealistically slow temperature schedule is used.

Implementation details are given in the appendix, Section A.1.2.

### 2.4.4 Hybrid methods

Each feature selection method has its own advantages and disadvantages. By combining complementary methods, we may be able to enhance the advantages and reduce the disadvantages.

---

<sup>6</sup>I.e. a change that increases the error of the model

Sequential forward selection is deterministic, and will rapidly perform a greedy search, but owing to its failure to reliably find combinations of features, it is often sub-optimal. Stochastic alternatives, such as simulated annealing, perform a wider search, but still, in practice, provide no guarantee of finding the global optimum, or even of finding a local optimum.

If we perform a single forward-backward sequential sweep (i.e. floating sequential selection) periodically during a stochastic search process, we can take advantage of greedy optimisation as well as stochastic optimisation. One approach is to combine simulated annealing with a floating sequential search, so that regularly during the search, a local greedy hill climber is applied to the current estimate. Reeves and Höhn [104] describe a hybrid system combining a genetic algorithm<sup>7</sup> with a local neighbourhood search, and show good results when applied to graph-partitioning and job-shop scheduling. However, given that we are using a simple binary representation, with each feature being either present or absent, such local hill climbing may have less effect owing to the lack of gradient information. In fact, we must be careful not to place too strong an emphasis on the greedy stage. If we were to perform a greedy step after every stochastic step, the effect would be to negate the stochastic search almost entirely, and to return repeatedly to the locally optimal model found by greedy search.

Another option would be to use a genetic algorithm to identify good regions, and then perform an exhaustive search within this region. But as noted earlier, the benefits of an exhaustive search are lost if the global optimum is not in the region being searched.

#### 2.4.5 Degree of search exhaustion

Having discussed several search methods, we now turn to the question of how effective they are. Given  $N$  features from which to choose a subset, there are  $N$  single-feature models,  $N \times (N - 1)$  two-feature models, and so on. As noted earlier, this gives  $2^N$  solutions, making an exhaustive search impossible. The question is what is the probability of finding the optimal solution? Given that we don't know this solution in advance (obviously), we can start by considering the probability of evaluating any given solution. The probability of not selecting some particular feature, at least once, if we build  $m$  single-feature models is:  $\left(\frac{N-1}{N}\right)^m$ . For example, the probability of never selecting feature 1 out of 65 within 1000 iterations<sup>8</sup> is  $(64/65)^{1000} = 1.85 \times 10^{-7}$ . This is so small that we can be reasonably certain that we will evaluate every single-feature model, even with a random search.

More generally, the probability of not selecting some particular combination of  $l$

---

<sup>7</sup>Genetic algorithms are population-based searches inspired by Darwin's theory of evolution.

<sup>8</sup>Here, one iteration refers to the evaluation of one feature subset, however it is generated.

features from  $N$ , after  $M$  iterations, is:

$$\left(1 - \frac{l!(N-l)!}{N!}\right)^m$$

Thus, the probability of never selecting one particular combination of 5 features from 65, after 10,000 iterations is 0.9988. So even with a modest number of features and a large number of iterations, we are still extremely unlikely to evaluate every model.

The nature of this coverage means that we need only concern ourselves with searching for feature subsets containing several features, safe in the knowledge that very small subsets will be searched near-exhaustively. The corollary is that the results from single-feature search are likely to show the best possible single-feature model, whereas the multiple-feature models are likely to be sub-optimal. This could introduce an unwanted bias towards simple models. Note that the symmetric argument applies to models containing all (or nearly all) the features, where there may be an unwanted bias towards very large models. It is the “middle” of the search space, with models containing approximately half of the available features, that may not be searched effectively.

The figures given above represent completely random feature selection, and the search methods used here are not completely random. Thus, for example, forward sequential selection will only cover a minuscule fraction of the space, but is biased towards locally good models; if this bias is suitable for the given search space, it will perform well despite its partial coverage. The same applies to any heuristic search method: the percentage covered is unimportant, *if* the best areas are searched. One advantage of simulated annealing is that it attempts “importance sampling” — features that seem to be useful to the current model, tend to be incorporated within future models. Thus, more important areas of the search space tend to be sampled more frequently than unimportant areas.

## 2.5 Avoiding overfitting

Intuitively, when we fit a model to a sample of data, we want to minimise the residual error. This can, in fact, always be reduced to zero if we use a sufficiently complex model (e.g. an RBF network with a sufficient number of kernels [63][15, p.167–169]). But this is only justified if we can guarantee that the data are noise-free. Any data set based on measurement will inevitably have some degree of noise, but we are interested in the underlying distribution, and not the noise itself. Overfitting can be defined as producing a model that fits the training sample but which fails to fit the underlying distribution of data, because the model is too complex. This leads to poor generalization to novel examples. Overfitting is often caused when the training data sample is distorted by

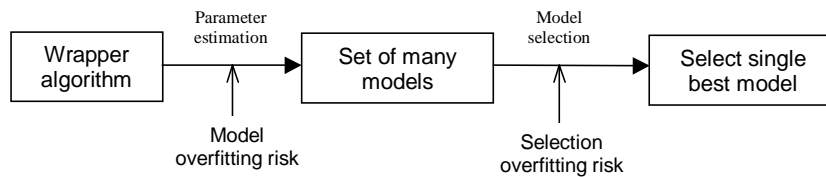


Figure 2.5: Two stages of overfitting with wrappers

noise: when a model fits the given noisy sample too closely, it will not fit the underlying distribution. Any error value associated with a model can only be an estimate of the true error. By selecting the model with the lowest *estimated* error, we may not be selecting the model with the lowest *true* error.

If we overfit during feature selection, we will select too many features and produce a poor regression model. If we produce a poor regression model, we will make spurious predictions regarding consumer preferences, claiming more than the data actually supports.

In many cases, overfitting can be avoided by applying cross validation methods (e.g. Ripley [105, p.69–70]), where we estimate the generalization performance of the model by using some extra data that was not used in estimating the model parameters. We build a model using the training data set, use this model to estimate the target values on the extra data, and compare these estimates with the true values. By minimising the generalization error, we can be more certain that we are fitting the underlying function and not the random noise. In fact, the more we overfit, the greater the generalization error will become, even if the training error continues to decrease.

However, wrapper techniques used in feature selection introduce a second source of overfitting, owing to their two-stage process. Firstly, many models are built. Secondly, one of these is chosen as the “best” model. Figure 2.5 shows these two stages. Even if cross validation is used during the first “parameter estimation” stage to give an good estimate of each model’s generalization performance, simply selecting the model with the lowest cross validation error (the “model selection” stage) will tend to cause overfitting. It will select a model that happens to fit the sample most closely, rather than one that fits the underlying distribution. This problem can be avoided by introducing a second, outer loop of cross validation into the algorithm<sup>9</sup>. This risk of overfitting applies to any search method, including feature selection using filters. But as these typically generate far fewer models from which to select, the bias is less likely to be important.

A third source of overfitting is because of “algorithm selection”. Suppose we avoid both parameter estimation and model selection overfitting (for example, using the tech-

<sup>9</sup>Logically, one could attempt to avoid “selection” overfitting without attempting to avoid “model” overfitting, but this seems rather pointless.

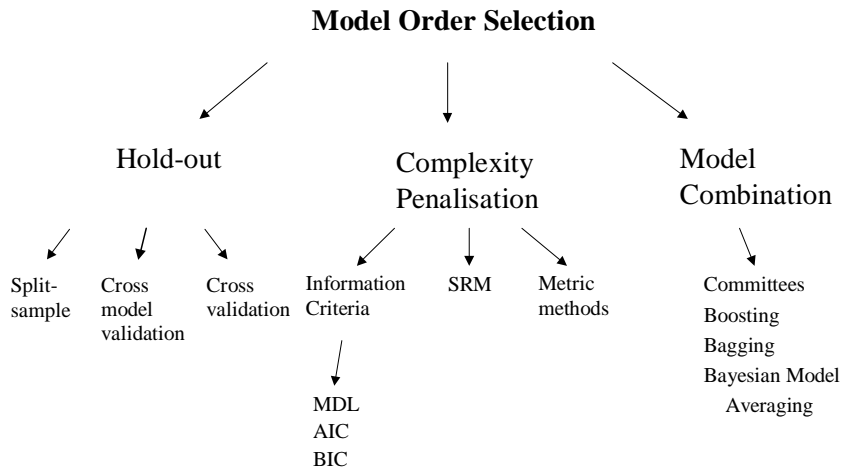


Figure 2.6: Model order selection hierarchy

niques described below in Sections 2.5.1–2.11), but then repeat this for many classes of models (RBF networks, support vector machines, decision trees and so on). If we then choose the model with the lowest error, there is a risk that this will overfit the training data. In this work, we reduce this risk<sup>10</sup> by limiting ourselves to just two classes of regression models, one linear and one non-linear.

A further source of overfitting is described by Salzberg [110] as being caused by “community experiments”, based on shared data sets, such as the UCI repository [16]. If many people use the same sets of data to compare the same algorithms, then even if there is no difference between the algorithms, one would still expect 5% of the researchers to find results significant at the 5% level — leading to a Type I error<sup>11</sup>. Although we do use UCI data in some experiments, most of the work presented here uses food data sets that are not in the public domain, and have therefore not been “overanalysed” by the machine learning and statistics communities.

The problem of selecting a model that is complex enough to model the data, without being so complex as to overfit, is the problem of *model order selection* [15, p.371]. Common examples of this are selecting the order of a polynomial (quadratic, cubic etc.) during curve-fitting or selecting the number of kernels in an RBF network. A number of techniques have been developed over many years; we show several in Figure 2.6 and discuss them in Sections 2.5.1–2.11.

<sup>10</sup>without entirely eliminating it

<sup>11</sup>A type I error occurs when we reject the null hypotheses incorrectly. A type II error occurs when we accept the null hypothesis incorrectly.

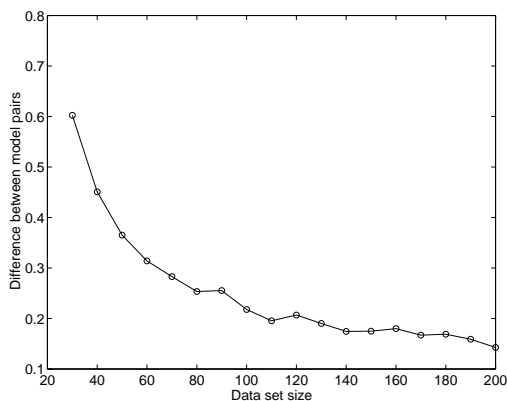


Figure 2.7: Bias when splitting data set

### 2.5.1 Hold out methods

Hold out methods use separate sets of data to estimate the model parameters and to estimate the generalization error (e.g. Hjorth [66, Ch.3]). The simplest approach is to split the data once, using the first part (the “training set”) to build the model(s) and the remaining part (the “test set”) to measure the generalization error. This can cause problems if the two sets are not identically distributed, because the error estimate will be inaccurate. This problem becomes progressively worse with smaller data sets.

To demonstrate this, we can create some artificial data, allowing us to vary the size of the training set. We define  $\mathbf{X} \sim N(0, 1)$ , so that  $\mathbf{X}$  is an  $n \times m$  matrix, consisting of  $n$  records with  $m$  features. Let  $\mathbf{w}$  be a random weight vector, of size  $m \times 1$ . Let  $\mathbf{Y} = \mathbf{X}\mathbf{w} + \epsilon$ , where  $\epsilon \sim N(0, 0.1)$  is a Gaussian noise term. We then split  $\mathbf{X}$  into two equal halves, each with  $n/2$  records and  $m$  features. We build two linear regression models, each with  $m + 1$  parameters, and define the distance between these models as the root mean square deviation between the parameter vectors. Figure 2.7 shows how the distance between pairs of models drops as we increase  $n$ , the size of the initial data set<sup>12</sup>. Thus, even though the two halves of the data set are drawn from the same underlying population, they lead to very different regression models, showing that the samples suggest substantially different distributions. The large variance between small samples causes errors when using a hold out sample to estimate a model’s accuracy.

A closely related method splits the data into three sets, training and test as before, and with an extra validation set (e.g. Bishop [15, p.373]). During iterative training algorithms (such as backpropagation, commonly used for neural network training), the validation set is used to stop training before overfitting occurs. Again, with a small data set this is of little use, because all three sets will have quite different distributions.

Another widely used hold out method is  $k$ -fold cross validation (e.g. Hjorth [66,

<sup>12</sup> $n$  varies from 30 to 200, with  $m = 15$  features.

p.27]). Here the data is split into training and test sets as before, and the sample generalization error is calculated. Then the data is re-split into different training and test sets, and used to produce a second estimate of the generalization error. By repeating this  $k$  times, and placing every record in a test set exactly once, then we can obtain the average of the sample generalization errors. This gives an unbiased estimate of the model's true generalization error. The most extreme version of this is leave-one-out cross validation, where each data point in turn is used as a test set (of size one), with the bulk of the data used for training. Thus, with  $N$  records,  $N$  models are built, each using  $N - 1$  training records.

By calling cross validation an “unbiased” estimate of the generalization error, we mean that on average it produces the best possible estimate, given the training data. In some cases, it will overestimate the error; in other cases it will underestimate the error. It is unbiased with respect to the training data, but not necessarily unbiased with respect to the underlying distribution. Therefore, if we build a large (but finite) number of models, each with its own cross validation error, and choose the model with the lowest estimate, we are likely to choose the model whose error was underestimated the most, and not the model with the lowest generalization error. This is likely to cause overfitting. We can avoid this effect using cross *model* validation, to which we now turn.

Hjorth [66, p.42] describes cross model validation, an algorithm which performs two nested cross validation loops. The generalization error of the best model of each possible model size is estimated. This is used to select the optimal model size. Then a separate cross validation loop is performed to optimise the parameters in a model of the given size, and to estimate the generalization error. In feature selection, this means first selecting the number of features, and then, as a separate stage, selecting which feature subset of the given size is optimal. This avoids both stages of overfitting labelled in Figure 2.5. Hjorth's description includes a loop over *all* models of a given size, which in the current work is infeasible. Instead, we can replace this exhaustive search with a stochastic search, such as simulated annealing, outlined in Section 2.4.3. Figure 2.8 shows the cross model validation algorithm.

Ng [91] describes a broadly similar approach, namely cross validated cross validation. He describes how overfitting may be caused by either building too complex a model, or else by building too many models. These correspond with the two stages of Figure 2.5. He argues that if we select a hypothesis that fits a sample of noisy data perfectly, it will necessarily fail to fit the underlying distribution; if a hypothesis fits the noisy sample less than perfectly, there is a greater chance that it will fit the underlying distribution optimally. He then describes a novel way to estimate how many hypotheses can be built, such that the model with the lowest error is best, without it overfitting

Given  $n \times m$  data set,  $D$ , with  $n$  records and  $m$  features, and a feature selection routine FS, do:

- 1 Initialise  $CMV$ , a  $m$ -dimensional vector of zeros
- 2 For  $i = 1$  to  $n$ 
  - 2.1 For  $j = 1$  to  $m$ 
    - 2.1.1  $h_s = \text{FS}(D_i, j)$
    - 2.1.2  $CMV(j) = CMV(j) + \text{err}(h_s, d_i)$
  - 2.2 next  $j$
- 3 next  $i$
- 4  $OMS = \min_j CMV$
- 5  $h_o = \text{FS}(D, OMS)$

where:

- $D_i = D \setminus d_i$ , i.e. the data set with  $i^{\text{th}}$  record removed;
- $OMS$  is the estimated optimum model size;
- $\text{FS}(D, S)$  is a function that performs feature selection using data  $D$  constrained to use exactly  $S$  features, and returns a single regression model;
- $\text{err}(h, d)$  is the error for model  $h$  estimated from test set  $d$ ; and
- $h_o$  is the final model selected.

Figure 2.8: Cross model validation algorithm, from Hjorth [66, p.42]

(the “best of  $n$ ” approach). From this, the percentage of models that are unlikely to overfit can be calculated; we can then build a very large number of models, sort them into descending order of estimated errors, and choose from the appropriate percentile.

One assumption behind Ng’s work is that the hypotheses are generated *i.i.d.* (independently and identically distributed). If we use the same sample of data to estimate the parameters of each model, then the resultant models will not be independent, so Ng suggests randomly sampling the hypothesis space. Any use of data to estimate parameters will introduce correlations between models, and hence destroy the required independence assumption. Given that even a directed search across the feature space is unlikely to find the optimum, a purely random search across both feature space and parameter space will not be productive.

### 2.5.2 Complexity penalisation

Rather than use extra data to estimate generalization errors, we can use regularisation to limit the complexity of the original model with respect to the size of the training set. Instead of minimising error (training or testing), we now minimise the sum of error and complexity, where “complexity” is typically a function of the number of free parameters being optimised by the model and the number of records being used to estimate these parameters. Examples of this include information criteria, such as Minimum Description Length (MDL) and Akaike’s Information Criteria (AIC), and structural risk minimisation (SRM, derived from statistical learning theory).

Historically, one of the first such complexity penalties was defined by Akaike and named “An Information Criteria” or AIC [1]. This defines the complexity term as twice the number of free parameters. While this gives an unbiased measure for the correct model, it gives a biased score if one model is selected from many (as discussed by Hjorth [66, p.46]). This is related to the “selection stage” of overfitting in Figure 2.5. In the current work, AIC will therefore tend to select models with the wrong number of features. Given sufficient data, the “Bayesian Information Criteria” or BIC described by Schwarz [117] gives an unbiased estimate and will therefore produce the “correct” model. However, this is an asymptotic result, and with very small data sets, the technique is unreliable. Rissanen’s Minimum Description Length (MDL) principle [106] was developed independently, but produces exactly the same result as BIC (except that it is negated). We use the MDL principle in Section 3.6 to estimate the number of clusters present in preference data. In Section 2.9 we use it to estimate the optimum number of features for a linear regression model, but with little success due to the lack of records.

Vapnik [132] describes structural risk minimisation (SRM), which is a principle derived in statistical learning theory, and, among other things, specifically deals with

small data sets. A complexity penalty in terms of the number of records and the complexity of the model (as defined using the Vapnik-Chervonenkis or VC dimension) is added to the empirical error (or “risk”) to avoid overfitting. Quadratic (or sometimes linear) programming is then used to solve an approximation to the resultant problem. This is the principal motivation behind support vector machines.

Much of the work in the SRM field provides bounds on probable errors. In principle, a bound could be defined which specifies the minimum number of records required to build a model of a given complexity, to a given accuracy and with a given probability. This is an example of “probably approximately correct” or PAC learning, as defined by Valiant [130]. Such bounds have been published for the classification case, the focus of much current research, but not for regression. For example, Christianini and Shawe-Taylor [33, pp.70–74] define an upper bound on linear regression error using (among other terms) the number of records, the radius of a sphere containing the data, and the degree of acceptable error, but crucially, not the number of features nor any equivalent measure. Furthermore, many of the published bounds are very loose, and therefore of limited value when trying to estimate the optimal number of features in practice.

Bradley and Mangasarian [17] describe the use of a particular type of support vector machine to perform feature selection while building a classifier. Rather than just minimising the weighted sum of error and complexity, they introduce a third term corresponding to the number of features, and a corresponding new weight parameter. They estimate the value of this new parameter using a validation hold out set. As argued above, hold out validation is unlikely to be successful when only a small data set is available.

Schuermans [115] proposes metric based model selection (MBMS) an alternative approach to model order selection. This requires access to a small set of labelled data and a separate set of unlabelled data<sup>13</sup>. A sequence of models of increasing complexity is built using the labelled data. The distance between each successive pair of models is then estimated using the unlabelled data. This is calculated by presenting both models with the unlabelled data, and calculating their predictions. Assuming each model has one output, then each set of predictions is a vector, and the distance between the two vectors (and hence models) is calculated using the Euclidean distance metric. The distance between a model and the underlying distribution is simply the root mean squared error, which is also a Euclidean distance measure. If the triangle formed between two models and the true distribution is invalid (i.e. if one length is greater than the sum of the two other lengths) then the more complex of the two models is

---

<sup>13</sup>In this context, the preference of a product is the label that we are trying to predict from the sensory data. Unlabelled data therefore describes products that the sensory panel has analysed, but which were never presented to a preference panel. This data is often available, such as the premium meat data sets.

assumed to be overfitting. The estimated distances suggest that the more complex model appears closer to the underlying distribution than it actually is – i.e. the error is underestimated and the model is overfitting. The last model in the sequence that does not violate this triangulation is considered the most complex model allowed without overfitting.

Although Schuurmans paper is based on selecting the order of a polynomial, we believe that the same approach can be taken to calculating the number of features that can safely be used. We therefore propose a semi-supervised feature selection algorithm in Section 2.8, p. 62.

The benefits of semi-supervised learning are felt most strongly when only a very limited set of labelled data is available. Given a “sufficiently large” set of labelled data, the extra information contained in any unlabelled data is unlikely to lead to models that are significantly more accurate. There are many instances when assigning a label to a record is the most expensive aspect of data gathering, such as labelling the objects in a picture [5]; rating the “interestingness” of a news article [92]; or describing preferences of different foods.

Besides semi-supervised feature selection (Section 2.8), we will also discuss two other forms of semi-supervised learning later: EM regression (Section 2.9), and a novel semi-supervised ensemble algorithm (Section 2.11). The meat sensory data contains 26 records with no corresponding preference scores, allowing the application of semi-supervised methods. In principle, the sensory panel could be presented with many more products, which would probably improve semi-supervised learning results. However, the data used here was not expressly gathered for such analysis. Results shown in Section 2.11.6 suggest that this relatively small extra unlabelled sample may not be sufficient to improve significantly the supervised learning results achieved without the extra records.

### 2.5.3 Model combination

A further method to reduce the risk of overfitting is model combination. This works by forming committees of several models, rather than selecting only one. A committee tends to achieve a lower error than the mean error of its constituent members, if the errors of each of the models are sufficiently uncorrelated (e.g. [97], [15, p.364]). This will be true if the members have been trained using different (possibly overlapping) sets of training data. The members of the committees should be uncorrelated so that each is accurate in a different section of the underlying distribution effectively, and taken together, they accurately model the whole distribution. Furthermore, if each member makes different mistakes, then taking the average should smooth out the errors and produce estimates closer to the underlying distribution.

Perrone and Cooper [97] show that a committee with  $M$  members can theoretically

have an error  $M$  times smaller than the average error of the members. However, this dramatic result only holds if the errors associated with committee members are completely uncorrelated. In naïve supervised ensemble learning, the ensemble members will tend to be highly correlated because each member is trained using the same data. Therefore, many ensemble methods are designed to minimise the error correlations, including bagging and boosting which are now described.

Breiman’s bagging algorithm [19] is a popular ensemble training method, where repeated bootstrap sampling from the training data is used to build a committee of learners. A new training set is created for each model, by randomly sampling with replacement  $N$  examples from the given data set of size  $N$ . Thus, in each training sample used, some records may be missing and others duplicated. This tends to produce distinct data sets, and hence different models, which between them cover the entire problem space. Bagging has been shown to be effective on weak, unstable learners<sup>14</sup>, including decision trees [19] and neural networks [94]. We use a form of bagging in our new semi-supervised ensemble learning, presented in Section 2.11.

Freund and Schapire [49] describe a boosting algorithm, which creates a series of weak learners and combines them to form a committee. Each learner is trained on the entire training data set, and each point in the set is assigned an individual weight. Each model is then penalised according to the weights associated with the points that it models inaccurately. During learning, data points that are modelled inaccurately are given increasingly large weights, which forces future learners to model them more accurately. Thus, when the committee of learners is formed, all of the data points should be modelled accurately by at least one weak learner, and therefore the committee’s final vote should be accurate over the entire problem space.

Krogh and Vedelsby [76] describe combining different neural network architectures into an ensemble. They derive a bias-variance trade off<sup>15</sup> for ensembles in terms of generalization errors of the ensemble members and the *ambiguity* of the members. Ambiguity is a measure of the disagreement of the members, and they show that higher ambiguity tends to produce lower generalization errors. The paper also demonstrates how unlabelled data can be used to estimate ambiguity, by measuring the correlations between ensemble members. This can improve the estimate of the generalization error.

Opitz [93] describes building an ensemble classifier, where each member is allowed to use a different set of features. A genetic algorithm is used to search the feature

---

<sup>14</sup>“Weak” refers to models with a high generalization error. “Unstable” refers to models that are very sensitive to small changes in the training data, which is particularly a problem with very small data sets [119].

<sup>15</sup>Bias is a measure of how well a model fits the underlying distribution, averaged over all samples from that distribution. Variance is a measure of how the accuracy of a model varies as the samples from the distribution vary. The expected error of a model is:  $(\text{bias})^2 + (\text{variance})$ . See Bishop [15, p.333–337].

sets of all the models simultaneously, and is designed to maximise both accuracy and diversity. This allows several useful feature sets to emerge at the same time. The results compare favourably with other ensemble methods (specifically bagging and AdaBoost, a common form of boosting). In the current work, we are trying to identify important drivers, as well as build accurate models. Allowing each member of an ensemble to select features independently may mask the importance of drivers. It may be possible to analyse further the ensembles produced in this way, and perform post hoc feature selection on the final ensemble, but this has not been attempted to date.

## 2.6 Evaluation of feature sets

We consider two possible methods of evaluating the effect of model order selection on choosing the number of features: cross model validation and resampling. Cross model validation was discussed earlier in Section 2.5.1, so here we discuss resampling.

To evaluate feature selection methods for small data sets, we can (perhaps counterintuitively) start with a large data set (artificially generated, or gathered from some other source), and repeatedly sample from it. Suppose we want to investigate the efficacy of feature selection given  $n$  (e.g. 16) records. We start with a large set of data with  $N \gg n$  records, from which we randomly select  $n$  records<sup>16</sup>. We perform feature selection on this sub-sample as if that were all that was available, estimating the required model size as well as selecting the features. We can then use the remainder of the large data set to evaluate the model, by measuring its generalization error. This result can be compared with feature selection results based on the entire data set, or on alternative methods using the same sub-sample. This form of resampling is inspired by the bootstrap (described by Efron and Tibshirani [40]), a computationally intensive method for estimating parameters. In bootstraps, a series of training sets are created by sampling with replacement from the original set, so that records may occur more than once<sup>17</sup>. These training sets can then be used to estimate statistical parameters of the underlying distribution.

Unfortunately, this is extremely unstable for feature selection: the main contribution to the generalization error estimate is the sample of data selected, rather than the feature selection (or regression) algorithm used (see Section 2.5.1). This leads to two alternatives. Firstly, we could accept the high variance, and simply compare the difference in generalization errors for each sample, and assume that repeated sampling will remove any bias to one model or another inherent in any given sample. Or secondly,

---

<sup>16</sup>We choose a large  $N$  to allow us to build models which generalize well, and a small  $n$  because the food data sets we are modelling have very few records. We are trying to estimate the generalization performance of models when we only have very few records available.

<sup>17</sup>As with bagging: the word “bagging” is a contraction of “bootstrap aggregating” [19]

we could choose the samples in some non-random fashion, such as choosing the  $n$  most widely separated points [73], after testing that they are reasonable estimates of the entire distribution. This instability is especially a problem when  $N$  is much larger than  $n$ , because then the smaller data set will tend to represent a different distribution than the larger, making generalization hard to estimate. (See Section 2.5.1 and particularly Figure 2.7.)

## 2.7 Results on food data sets

Having discussed the problems of regression, feature selection and overfitting, we now apply several of the techniques described, and compare and contrast the results. Table 2.1 compares a number of feature selection routines when applied to the meat sensory data set,  $S_m$ . The data set used consists of 65 sensory features, which are used to predict the mean consumer preference of each of 16 products. Each experiment used a different feature selection technique, wrapped around a regression algorithm. The errors quoted are the root mean square errors. Table 2.3 defines the abbreviations used and in which section each technique is described. Table 2.2 shows the same feature selection methods applied to the vegetable data,  $S_v$ , excluding those techniques that require an unlabelled data set.

Table 2.1 is ordered from the smallest to greatest degree of validation and avoidance of overfitting. Each pair of rows compares linear regression with RBF regression. The top pair shows simple forward feature selection with no attempt to avoid overfitting. All 16 records are used to greedily select the best features to build a model. The best models found have many features and extremely low errors. These results are suspicious on two counts. Firstly, the regression models require estimation of many parameters from only a few records, because they have selected so many features. Secondly, the errors are extremely small, and we have every reason to believe that the data set is noisy. The source data set contains consumer preferences expressed on a scale of one to nine; estimating these values with a claimed accuracy in the region of  $10^{-11}$  seems over precise.

The next six rows in the results table (labelled “M”) show a first attempt at model validation, namely using cross validation on parameter estimation, to avoid the “model” overfitting shown earlier in Figure 2.5. Most of these results also have low estimated errors and many features. The errors are all calculated using leave-one-out cross validation, known to give an unbiased estimate (e.g. Hjorth [66]). However, as noted in Section 2.5, selecting one model from amongst many introduces its own bias, and it is this which suggests that these estimated error scores are gross underestimates. As before, these models are suspiciously large and “accurate”, although less so than before.

Feature selection technique	Regression model	Num. of features	Num. of parameters	Test error	Overfitting stages avoided
FSS	Linear	14	15	$5.77 \times 10^{-7}$	-
FSS	RBF	6	42	$1.34 \times 10^{-12}$	-
FSS + CV	Linear	8	9	0.0600	M
FSS + CV	RBF	4	30	0.0339	M
SA + CV	Linear	14	15	$1.07 \times 10^{-4}$	M
SA + CV	RBF	11	72	0.166	M
CBFE + CV	Linear	9	10	0.415	M
CBFE + CV	RBF	8	54	0.396	M
FSS + CV + MCST	Linear	0	1	0.914	M, S
FSS + CV + MCST	RBF	0	1	0.914	M, S
ADJ	Linear	3–6	4–7	0.435	M, S
ADJ	RBF	1	12	0.909	M, S
SA + CMV	Linear	3	4	0.158	M, S
SA + CMV	RBF	2	18	0.117	M, S
Exhaustive CMV	Any	N/A	N/A	N/A	M, S

Table 2.1: Feature selection results — meat data. Key in Table 2.3.

Feature selection technique	Regression model	Num. of features	Num. of parameters	Test error	Overfitting stages avoided
FSS	Linear	15	16	0.00646	-
FSS	RBF	14	90	$1.18 \times 10^{-11}$	-
FSS + CV	Linear	5	6	0.536	M
FSS + CV	RBF	2	18	0.886	M
SA + CV	Linear	14	15	0.0311	M
SA + CV	RBF	5	36	0.147	M
CBFE + CV	Linear	2	3	1.02	M
CBFE + CV	RBF	7	48	1.18	M
FSS + CV + MCST	Linear	1	2	0.325	M, S
FSS + CV + MCST	RBF	0	1	1.10	M, S
SA + CMV	Linear	3	4	0.115	M, S
SA + CMV	RBF	7	48	0.231	M, S
Exhaustive CMV	Any	N/A	N/A	N/A	M, S

Table 2.2: Feature selection results — vegetable data. Key in Table 2.3

Abbreviation	Method	Section
FSS	Forward sequential selection	2.4.2
CV	Cross validation (for parameter estimation)	2.5
SA	Simulated annealing	2.4.3
CBFE	Correlation-based feature elimination	2.3
MCST	Monte Carlo significance test	2.4.2
ADJ	Metric-based model selection — adjusted “ADJ” test	2.5.2
CMV	Cross model validation (for size estimation and parameter estimation)	2.5.1
M	Model building (parameter estimation) overfitting avoided	2.5
S	Model selection (size estimation) overfitting avoided	2.5
N/A	Not available owing to computational complexity. These results would take millennia to calculate on a typical Pentium III PC.	2.4.1

Table 2.3: Key to Tables 2.1-2.2

The next six rows (labelled “M, S”) show methods that avoid both stages of overfitting, by incorporating cross model validation or alternatives, described earlier. The estimated errors here are higher, but are less likely to be underestimates of the true generalization error. Similarly, the estimated number of features is usually much smaller, often suggesting that we can safely use a very small number of features when building models.

The Monte Carlo significance test (MCST) is a stopping criteria for forward sequential selection (Section 2.4.2, p. 44). For each feature addition being tested, the change in error is calculated with respect to the previous model, i.e. a model with one less feature. For each possible number of features, a normal curve is fitted to the distribution error changes. If the addition of a feature is found to decrease the error of the model by a significant amount, then the feature is added, and the feature selection process continues. If at any stage the biggest decrease in error is *not* significant, then no further features are added to the model, and the feature selection process terminates.

MCST suggests that even adding in one feature produces an insignificant reduction in error (at the 5% level), resulting in a model with zero features. One explanation for this extreme result is that the errors may not follow a normal distribution sufficiently closely to use this test reliably. Nevertheless, the corresponding error scores are useful as an indication of how accurate all the other models are.

The final row, included for the sake of completeness, shows the computationally infeasible exhaustive search. This, with cross model validation, is the only method

guaranteed to find the optimal result<sup>18</sup>, which is not given for obvious reasons (see Section 2.4.1). The “guarantee” that such an exhaustive search would produce optimal results assumes that cross model validation perfectly avoids overfitting.

Table 2.2 gives corresponding results for the vegetable data, and follows a similar pattern. As we move down the table, we avoid overfitting more carefully, and the estimated errors rise as the number of features used falls. Again, a simple application of forward sequential selection produces complex models which appear to overfit. The use of cross validation by itself may not be sufficient: the simulated annealing results still select many features, although the RBF variant seems plausible. The cross model validation results show reasonable error scores and the method selects only three features with linear regression, suggesting that the models are not overfitting, although the RBF version does select many more features. The ADJ test is not possible with this data set, because unlike the meat data set, there is no extra unlabelled data.

Overall, stochastic search methods (simulated annealing) with cross model validation seem to produce the best results. The cross model validation avoids overfitting, while the search algorithms seem to allow useful combinations of features to emerge. Note that the data sets are all scaled in the range 1–10, so a generalization error of around 0.1 is perfectly acceptable, while an error around  $10 \times 10^{-12}$  is extremely small. In most of the meat data set results, the non-linear RBF networks produce slightly lower error scores than the linear regression models, although the opposite is true with the vegetable data set.

The actual feature sets selected by the various search algorithms are not reported here, for reasons of commercial confidentiality. In general, different sets were found each time the stochastic algorithms were used, even if the final regression errors were (almost) identical. This suggests that there is no single optimum set of features, but rather that several sets exist that are all, effectively, optimal. Many of these sets differ only by a feature or two, and in most cases, these differing features are clearly related, perhaps measuring some particular flavour and the corresponding odour, or measuring the “first bite” texture to the “chewiness”. Nevertheless, a brief examination of several alternate sets of features by an expert can reveal similarities, allowing a concise description of the key descriptors to be derived.

## 2.8 Semi-supervised feature selection

We now describe a novel semi-supervised approach to feature selection, extending Schuurmans’ work on model order selection [115] described earlier (Section 2.5.2). In our new method, we use the labelled data to perform conventional forward sequential selec-

---

<sup>18</sup>Optimal with respect to the available data.

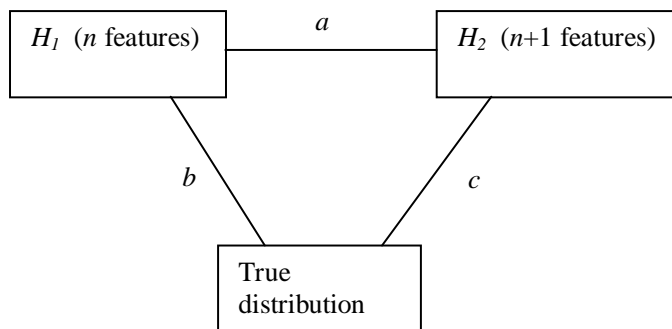


Figure 2.9: Semi-supervised feature selection

tion (Section 2.4.2) and build a series of models with an increasing number of features. We then use the unlabelled data to estimate the distances between each adjacent pair of models. When the triangulation rule (explained below) is broken, we know that too many features have been selected, so we can limit the model size accordingly.

### 2.8.1 Method

Given  $n$  features, we build a sequence of  $n + 1$  models, with between 0 and  $n$  features, using forward sequential selection. Each model is then presented with the  $m$  unlabelled sensory records, and the corresponding predictions are recorded. The Euclidean distance between these successive  $m$ -dimension vectors is then calculated to estimate the distance between each adjacent pair of models ( $a$  in Figure 2.9). The lengths of the other two sides of the triangle ( $b$  and  $c$  in Figure 2.9) are simply the estimated errors of the two models,  $H_1$  and  $H_2$ . If  $c > a + b$ , then, following Schuurmans argument, the more complex model,  $H_2$ , is assumed to be overfitting.

This basic triangulation test is extended by Schuurmans into an “adjusted distance estimate”, or ADJ. As well as using the unlabelled data to measure the distance between the two models, we can use the labelled data. This gives us a ratio between the two estimates in the local area, which can then be used to adjust the estimated distances between the models and the underlying distribution. Further details are given in the appendix, Section A.1.3. Although Schuurmans recognises this to be a “naïve assumption” [116], he also demonstrates that it outperforms the triangulation test, and many widely-used regularisation methods when learning from small labelled data set.

### 2.8.2 Results on meat data

We now compare forward sequential selection to the proposed semi-supervised feature selection. Figures 2.10–2.11 show the results when a linear regression model is used. The meat data used here has 16 labelled records and 26 unlabelled records in the

sensory set. We use one labelled record as a test set, giving 16 experiments. In each case, we use forward sequential selection (FSS) with cross validation and a maximum of 12 features, shown as white bars labelled “FSS” in Figure 2.10. For the semi-supervised version, we built a series of models, with between 0 and 12 features, again selected using FSS. The ADJ test was used to decide when too many features were being used. The error of the resultant model is shown as dark bars. The average errors were 0.435 for ADJ, and 0.347 for FSS. These are not significantly different according to a  $t$ -test ( $p = 0.184$ ). Note that although FSS is marginally better on average, it does appear to fail substantially on at least one occasion (sample 7), although this is a very small sample of results.

Figure 2.11 shows the number of features used in each case. Clearly, ADJ always selects fewer features than FSS, even when cross-validation is being used to restrict the number of features, and therefore to avoid overfitting. On average, ADJ uses 4.0 features; FSS uses 9.1. A  $t$ -test gives  $p \approx 0$ .

Note that the FSS errors quoted here are larger than those quoted in Table 2.1 earlier. This is because we are now carrying out 16 separate experiments, each one using cross validation on the training set of 15 records, with the error measured using the remaining one. Previously, we used all 16 records to build a single model, with ADJ to avoid overfitting.

Figures 2.12–2.13 make the equivalent comparison using RBF networks in place of linear regression models. The RBF networks tend to select fewer features than linear regression models, although the errors are higher. Figure 2.12 shows that across the 16 samples, ADJ tends to have a slightly higher error than FSS. The average errors were 0.909 for ADJ, and 0.723 for FSS. As with the linear models, Figure 2.13 shows that ADJ tends to select fewer features than FSS, in this case choosing just one feature every time.

These results suggest that when Schuurmans’ adjusted distance metric is applied to feature selection, it tends to select too few features, leading to poor generalization. This could be due to the size of the data sets: perhaps 26 unlabelled records are not enough to reliably measure the distances between models. Alternatively, it could be that using forward sequential selection to build a series of models leads to highly correlated outputs, making the ADJ test unreliable. Models that vary only in the *number* of features used may produce highly correlated outputs. For example, if we build a model using four features, and then build a second model with those same features plus one extra, then the two sets of outputs from these models are likely to be very similar. If this happens, then the triangulation test may fail to distinguish between the models, and so fail to recognise when one model is overfitting. We will then produce models with too many features, and these may overfit the data sample. Future work could

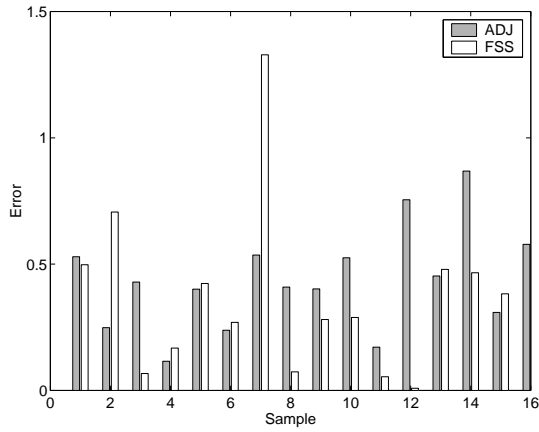


Figure 2.10: Semi-supervised feature selection — linear regression error for meat data

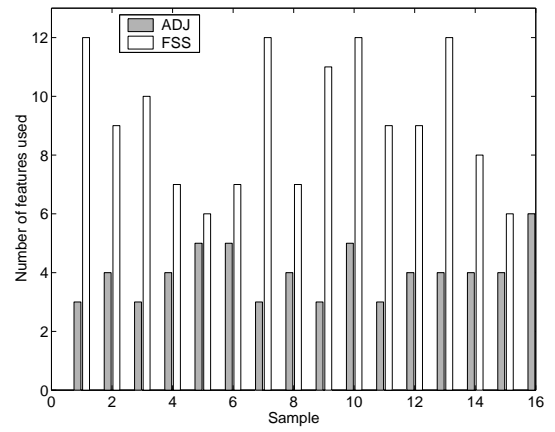


Figure 2.11: Semi-supervised feature selection — linear model size for meat data

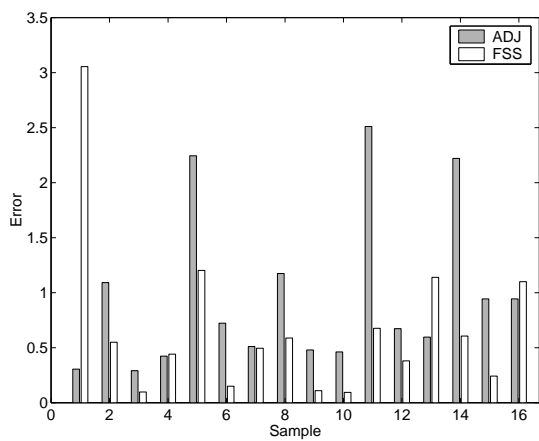


Figure 2.12: Semi-supervised feature selection — RBF error for meat data

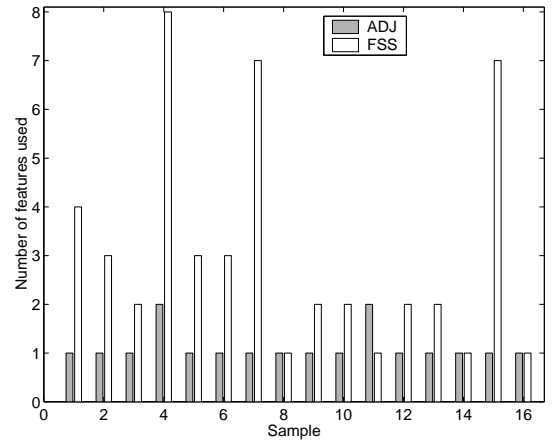


Figure 2.13: Semi-supervised feature selection — RBF model size for meat data

include using alternative feature selection routines with ADJ (Section 5.3.1).

## 2.9 Minimum description length for regression

The likelihood of a data set  $\langle \mathbf{Y}, \mathbf{X} \rangle$  given a regression model with parameters  $\theta$  is given by Baxter [8, p.129]:

$$l(\mathbf{Y}|\theta) = \prod_{i=1}^n \frac{1}{\sigma\sqrt{2\pi}} \exp\left(\frac{-(y_i - f(x_i))^2}{2\sigma^2}\right)$$

where  $n$  is the number of records, each with an associated error  $(y_i - f(x_i))^2$ , and  $\sigma$  is the standard deviation of the errors. This assumes that the errors are independent and have a Gaussian distribution. In fact, this formula is a simplification: Baxter follows Wallace and Boulton [133] and introduces extra values defining the accuracy of measurement both of the variables and of the model parameters. However, we can safely assume that these are constant across all models under consideration, and as we aim to maximise the likelihood (or, equivalently, to minimise the negative log likelihood), we can ignore these constant factors. The number of free parameters of a linear regression model with  $m$  features is  $m + 2$ : one for each feature, plus the bias term and the standard deviation estimate for the errors. The description length is the sum of the likelihood and the model complexity, the latter given by  $\frac{1}{2}(m + 2) \log n$ , with  $m + 2$  free parameters.

As a simple study, we build 10,000 linear regression models, each with a uniform random selection of between 0 and 14 features. Figure 2.14 shows the description length for linear regression models, as the number of features used varies. The minimum clearly corresponds to using the maximum 14 features. To understand why, Figure 2.15 separates the two components of error<sup>19</sup> and complexity. The error (solid line) drops sharply as we use more features, presumably due to substantial overfitting. The complexity penalty (dashed line) is too small to compensate for this, partly due to the lack of records. Given more records, the description length would increase, because of the  $\log n$  factor. If the models are overfitting, then increasing the number of records would probably lead to simpler models, and therefore a *lower* description length. Using a harsher penalty complexity could produce more useful results, but this is a subject for future work.

---

<sup>19</sup>negative log likelihood in this case

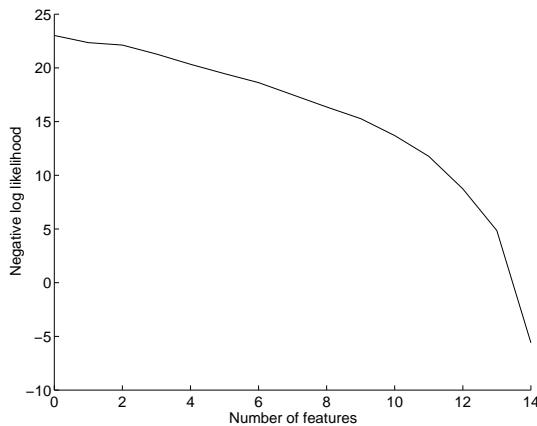


Figure 2.14: MDL scores for linear regression on meat data

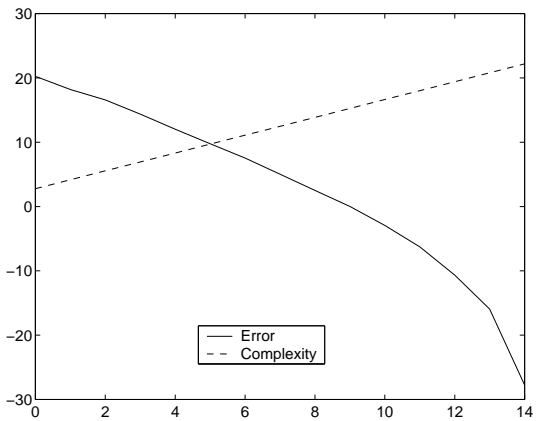


Figure 2.15: MDL scores for linear regression on meat data — error and complexity

## 2.10 The EM algorithm for regression

We have used a mixture of labelled and unlabelled data for feature selection (Section 2.8), and for building committees (Section 2.11). We now investigate the use of both types of data for building single (as opposed to a committee of) non-linear regression models.

Nigam et al. [92] describe a simple application of the expectation-maximisation (EM) algorithm for semi-supervised learning. They build a model using the labelled data, and use this to estimate the unknown labels. These estimated labels, along with the original labelled data set, are used to improve the model. By iteratively re-estimating the labels and re-building the model, they demonstrate a significant reduction in text classification error.

This is similar to the semi-supervised ensembles discussed earlier (Section 2.11), except that this time, no committees are formed but rather a single model is iteratively improved. Also, the labels are re-estimated, rather than randomly permuted. This version of the EM algorithm is shown in Figure 2.16.

Figure 2.17 shows the error as we repeated apply the EM algorithm to the meat sensory data, and thus combine the labelled and unlabelled data. An RBF network with eight basis functions was used, and the generalization error was estimated using leave one out cross validation. The six features used were selecting by forward sequential selection. The experiment was repeated 100 times, and the results averaged. The leftmost point on the graph shows the error before any EM iterations, and as the number of iterations increases, the error drops. The initial root mean squared error is 9.58; the final error is 1.48. The error drop is far from monotonic; after about five iterations, the system becomes unstable as we continue to change the estimated labels.

Given labelled data  $(x_l, y_l)$  and unlabelled data  $(x_u)$ :

1. Build regression model  $f_0 : x_l \mapsto y_l$
2. Use this model to estimate labels  $y'_u \leftarrow f_0(x_u)$
3. Iterate:
  - (a) Build new model  $f_i : (x_l, x_u) \mapsto (y_l, y'_u)$
  - (b) Re-estimate labels  $y'_u \leftarrow f_i(x_u)$

Figure 2.16: EM regression algorithm

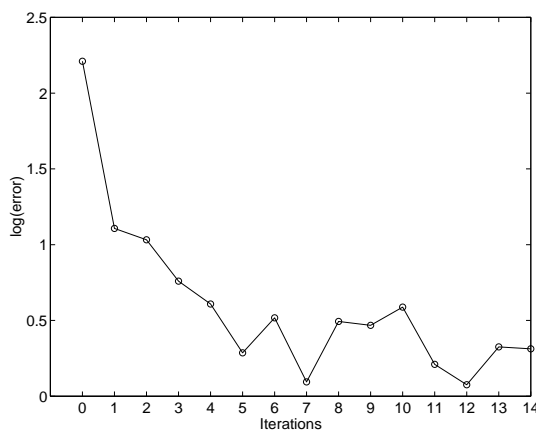


Figure 2.17: EM algorithm on RBF networks for meat data

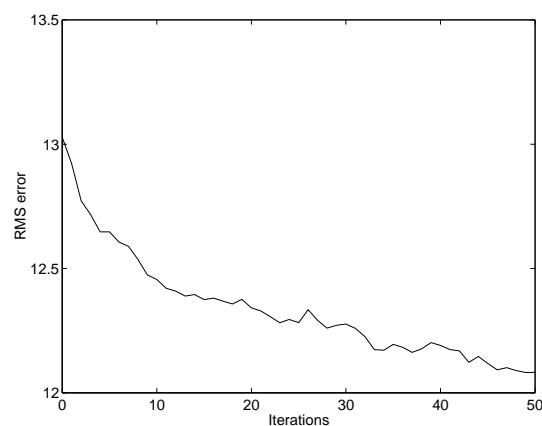


Figure 2.18: EM algorithm on RBF networks for small Boston sample

A simpler model (e.g. a smaller RBF network) would be more stable, but potentially less accurate. Note that the vertical axis shows the log of the error for clarity. For comparison, Figure 2.18 shows the equivalent results using the Boston Housing data. This shows the root mean square error, which drops (almost) monotonically, albeit slowly. This data is used in Section 2.11.1 to test semi-supervised ensemble learning.

## 2.11 Semi-supervised ensembles

We now describe a novel form of committee, which uses a combination of labelled and unlabelled data to build models that are both accurate and varied. To build a committee, we want to select a group of models that *i*) make accurate predictions for labelled records; but *ii*) disagree in their predictions for unlabelled records. This forces the errors to be less correlated, which allows more accurate committees to be formed, as we discussed in Section 2.5.3. Then, when the ensemble members make predictions, we can regard the disagreements between models as a form of noise, to be smoothed by

Given a set of labelled data  $(X_L, Y_L)$ , with each  $Y_i$  being the label corresponding to the descriptors  $X_i$ , and  $X_U$  a set of unlabelled data, drawn from the same distribution as  $X_L$ , build a committee of size  $s + 1$  as follows:

1. Estimate  $f_0 : X_L \rightarrow Y_L$
2. Use  $f_0$  to predict labels  $Y_U$  for  $X_U$ . i.e.  $Y_U = f_0(X_U)$
3. For each model,  $f_i, i = 1 \dots s$  do:
  - (a) Modify  $Y_U$  to form  $Y'_U$  by permuting the labels
  - (b) Estimate model:  $f_i : (X_L, X_U) \rightarrow (Y_L, Y'_U)$

Figure 2.19: Semi-supervised ensemble learning (SSEL) algorithm

averaging. Past approaches have either combined different types of model (e.g. Krogh and Vedelsby [76]) or used different (possibly overlapping) data sets for each model (e.g. boosting and bagging, described above).

The proposed approach explicitly forces the models to be distinct, by using data sets that are *contradictory*, rather than just different subsets of a larger set, or differently weighted examples. The approach is to build the first member of the committee using the labelled data, as in a standard supervised learning method. This model is then used to estimate labels on the unlabelled data. *These labels are then randomly changed*, as discussed below. Subsequent models are trained using both the original labelled data and the randomly re-labelled data. The algorithm is outlined in Figure 2.19. This is a variation on the EM algorithm for regression, described in Section 2.10.

By changing the estimated labels of the unlabelled data in step 3a of the algorithm, we are maintaining the marginal distributions (i.e. the independent distribution of each  $X$  variable) while disrupting the joint distribution (i.e. the distribution of  $X$  and  $Y$  together). This strange action requires some justification. Firstly, the joint distribution we are disrupting is only over the records that were originally unlabelled. Thus, we are altering (possibly poor) estimates rather than (more reliable) observations. Secondly, if the joint distribution is unchanged, the models will tend to be correlated, and that is exactly what we are trying to avoid. Thirdly, if we are going to re-label the data, the most sensible distribution from which to draw new labels is our best estimate of the underlying distribution, i.e. the predictions of a model that we have already built from the data. The approach used here is to permute the estimated labels with respect to the unlabelled  $X_U$  descriptors, so that each estimated label is no longer associated with the record from which it was derived.

In Figure 2.9 and the accompanying discussion, we described how Schuurmans distance metric [115] measures the distance between pairs of models using unlabelled

data. Our proposal changes the labels to maximise this distance, and hence produce a diverse committee. By forcing two models to make different predictions for each unlabelled record, the distance metric will tend to increase.

With reference to the algorithm of Figure 2.19, let us compare  $f_0$  and  $f_i$ ,  $i > 0$ . They should be approximately equally good at predicting  $Y_L$  — the labelled data, which is the basic performance measure — because they have both been presented with this data set during training. But when presented with  $X_U$ , one will attempt to predict  $Y_U$ , the other will attempt to predict  $Y'_U$ , and  $Y_U \neq Y'_U$  by definition. Thus,  $f_0$  and  $f_i$  are different (in the region of  $X_U$ ) but accurate (over  $X_L$ ). The same is true for any pair of models in the ensemble, if each  $Y'$  is a distinct permutation. Our motivation then, is to build a committee of diverse yet accurate models. The analysis of correlations between models in Section 2.11.3 contradict our expectations, although this technique is still shown to be effective.

This approach to semi-supervised ensemble building can be applied to bagging. In step 3b of the SSEL algorithm, we form a temporary data set by concatenating the labelled data with the randomly re-labelled data, and then uniformly select from this, with replacement, to form the actual training set. The resultant regression models are combined by taking the average of their predictions, as described by Breiman [19].

We now describe a series of experiments, initially using a public-domain data set as a benchmark, and then turning to the food data sets. In the following sections, we consider correlations between members of a committee (Section 2.11.3), the effect of committee size (Section 2.11.4), and the gains achieved using a committee (Section 2.11.5). In Section 2.11.6 we vary the sizes of both the labelled and unlabelled data set, to gain a clearer understanding of when the SSEL approach is useful. In Section 2.11.7, we describe equivalent results for the meat data set.

### 2.11.1 Semi-supervised bagging for Boston Housing data

We now describe experiments performed using semi-supervised bagging on a public domain data set. The Boston Housing data set (available from the UCI repository [16]) consists of 506 records describing properties in the Boston area. This larger data set allows us to investigate how the SSEL algorithm performs with different numbers of records, something that we cannot do using the food data alone. The target feature here is price, to be predicted from 13 other features. We randomly sample  $n_L$  records (20, 50 or 100 in the experiments below) from the 506 available to use as a labelled training set. We select a further  $n_U$  records (80 in these experiments) from which we discard the labels and treat them as unlabelled records. A further 100 records are used as a final test set. For each data set, we build three predictive models, each consisting of one or more RBF networks:

1. A single RBF network (“single” in the results sections).
2. A committee of 50 RBF networks, each trained by conventional (i.e. supervised) bagging (“bagging”).
3. A committee of 50 RBF networks, using semi-supervised ensemble learning, as described above (“SSEL”).

In all the committees, the first member was trained using all the labelled data. This eases comparisons between techniques by ensuring that all committees start from the same point. Note that this is just a special case of bagging with uniform selection. The ensemble output is the (uniformly weighted) mean of the members’ outputs. The number of basis functions in each RBF network was set at  $\lfloor 2\sqrt{n_L} \rfloor$ , when trained with  $n_L$  labelled records. This simple heuristic was selected to avoid the extra computational expense of using cross validation to determine the optimum model size, and also to reduce the risk of overfitting that this might cause. We discuss this heuristic further in Section 2.11.2.

For the small labelled set, we use  $n_L = 20$  labelled training records. Each RBF network had 8 basis functions. Table 2.4 shows the root mean square error scores for the three methods, averaged over 150 experiments. The standard deviations of these errors are also shown, as is the single largest error of the experiments. This clearly shows that SSEL outperforms both supervised bagging and a single network. A sign test [120] shows that the mean SSEL error is significantly lower than the mean bagging error ( $p = 1.96 \times 10^{-7}$ ) and significantly lower than the single model error ( $p = 3.99 \times 10^{-5}$ ).<sup>20</sup>

Some further points should be noted. Firstly, we are using a very small sample of data for training, so the error scores are appreciably higher than those typically published for this data set. For example, Drucker et al. [37] use support vector regression on a large sample from the Boston Housing set, and achieve a mean squared error of 7.2, as opposed to the *root* mean squared error of 9.3 quoted in the table. Secondly, each experiment uses a different sample of training data, so the standard deviation scores are large. Thirdly, both the single model and the supervised bagging model show very high “maximum error” scores (i.e. the worst performance across the experiments) compared with the semi-supervised version. This is reflected in the higher mean and standard deviation scores. Finally, the supervised bagging errors were higher than the single model errors.

For the medium labelled set, we use  $n_L = 50$  labelled training records. Each RBF network had 14 basis functions. Table 2.5 shows the average error scores for the three

---

<sup>20</sup>The sign test is used here because the distributions of error scores are not normal or symmetric, making more powerful tests such as the  $t$ -test inappropriate.

	Single	Bagging	SSEL
Mean Error	12.622	20.535	9.303
Standard Deviation	16.562	29.650	2.759
Maximum Error	177.701	210.197	35.668

Table 2.4: Semi-supervised ensemble results — small Boston sample

	Single	Bagging	SSEL
Mean Error	7.375	7.132	7.424
Standard Deviation	1.117	1.554	0.486
Maximum Error	13.652	13.989	9.797

Table 2.5: Semi-supervised ensemble results — medium Boston sample

methods over 150 experiments. A sign test shows that the supervised bagging errors are significantly lower than both the SSEL errors ( $p = 9.58 \times 10^{-10}$ ) and the single model errors ( $p = 1.78 \times 10^{-5}$ ). The single model is also better than SSEL ( $p = 0.0081$ ). The standard deviation of the errors is much lower for SSEL than for supervised bagging, and as before, the worst-case errors are also lower.

For the large labelled set, we use  $n_L = 100$  labelled training records. Each RBF network had 20 basis functions. Table 2.6 shows the average error scores for the three methods over 150 experiments. A sign test shows that the mean SSEL error is significantly worse than both the bagging error ( $p = 1.55 \times 10^{-22}$ ) and the single model error ( $p = 3.70 \times 10^{-4}$ ).

After these three sets of experiments, we can see that SSEL outperforms both conventional bagging and a single RBF network on these data sets, where the labelled set is very small. We consider this further in Section 2.11.6.

### 2.11.2 RBF network size heuristic

When using RBF networks, the literature typically advises us to perform cross validation to select the optimum number of basis functions (e.g. [15, p.371]). With such limited data as we have available, this may not be appropriate, as we discussed in Section 2.5.1. Furthermore, in the experiments we have described above, we are varying

	Single	Bagging	SSEL
Mean Error	6.230	5.807	6.441
Standard Deviation	0.559	0.449	0.416
Maximum Error	5.248	4.984	5.592

Table 2.6: Semi-supervised ensemble results — large Boston sample

the number of records forming the training set, which is a relatively unusual action. Instead, we designed and used a simple heuristic, specifying the number of basis functions as a function of the number of records in the training set. Hindsight suggests that this heuristic may select too many basis functions, especially with the small data sets being used. In light of the results in Tables 2.4–2.6, could we obtain lower error scores by using a more sensible heuristic? No doubt we could, but would these be *accurate* estimates of the errors? To avoid overfitting, we must use separate data sets to build the model and to estimate the error. If we use the results to guide the design of a better heuristic (and hence to build better RBF networks), then we should subsequently use a separate data set to test the new models. But of course, this extra data set is not available.

One solution to this problem would be to introduce another stage of the cross model validation algorithm. The standard version outlined in Figure 2.8 contains two nested loops: the inner loop selects the features for different sized models, while the outer loop uses the hold-out data to choose between these models. A third loop could be introduced, inside the current innermost loop, to vary the number of nodes in the RBF networks independently of the number of features being used. We discuss this further in Section 5.3.2.

### 2.11.3 Correlations and distances

By defining a Euclidean metric over the test data set, we can measure the distance between committee members, in the way proposed by Schuurmans [115] and described in Section 2.11. The test set is presented to the models, each of which predict a vector of outputs. The Euclidean distance between each pair of vectors is then calculated. Table 2.7 shows the average distance between each regression model and its nearest neighbour. Compared to supervised committees, the semi-supervised committee members are closer to their nearest neighbours. As the number of labelled training records increases, the bagging inter-model distances are reduced, tending towards the corresponding semi-supervised distances.

Similarly, we can measure the correlations between the test errors within each committee. Table 2.8 shows the average correlations between each pair of RBF networks in the committee, for the two ensemble methods and for the three sizes of labelled data set. Overall, the members of semi-supervised committees are much more highly correlated than their supervised counterparts. As the size of the labelled training set increases, the supervised models tend to become more highly correlated.

Note that these distance and correlation results contradict our original intentions, explained in Section 2.11. The SSEL algorithm was designed to produce ensembles whose members were further apart and less correlated than would be the case in con-

Labelled set size	Bagging	SSEL
20	11.7 (22.0)	4.60 (1.75)
50	4.90 ( 3.12)	3.79 (0.847)
100	3.64 ( 1.46)	3.48 (0.706)

Table 2.7: Semi-supervised ensemble distances — Boston data. Standard deviations are shown in parentheses.

Labelled set size	Bagging	SSEL
20	0.363 (0.164)	0.750 (0.204)
50	0.563 (0.128)	0.755 (0.110)
100	0.671 (0.0878)	0.754 (0.0822)

Table 2.8: Semi-supervised ensemble correlations — Boston data. Standard deviations are shown in parentheses.

ventional, supervised bagging ensembles. Despite this, our SSEL approach does outperform supervised bagging in some cases.

Ensembles are used because their errors tend to be lower than the mean error of their members. Analysis of the committees built here shows that in all cases, the committee error is lower than the mean member error, as expected. However, the errors of individual members for the semi-supervised case are far lower than for the supervised case, as we will show in Section 2.11.5.

#### 2.11.4 Committee sizes

Figures 2.20 – 2.22 shows the test error of the two types of committee as members are added. In each case, the first member of the committee was trained using all the available labelled data, i.e. 20, 50 or 100 records. All results shown are the average of 150 experiments, each with random samples from the Boston data set.

Figure 2.20 shows that with a very small labelled set, semi-supervised bagging consistently outperforms conventional bagging. After just five members are included, the semi-supervised error rate (solid line) levels off. In contrast, the supervised bagging (dotted line) performs badly, with the second and subsequent members *increasing* the error, before it drops, and levels off after about 35 members. The initial rise is because the first model uses all the labelled data, while the second and subsequent models use a bootstrap sample.

Figure 2.21 shows that with a moderate sized labelled set, and a sufficiently large committee, then supervised bagging outperforms semi-supervised. The increasing error of semi-supervised bagging, when three or more members are included, is presumably an effect of the random labels. This is discussed further in Section 2.11.8. Figure

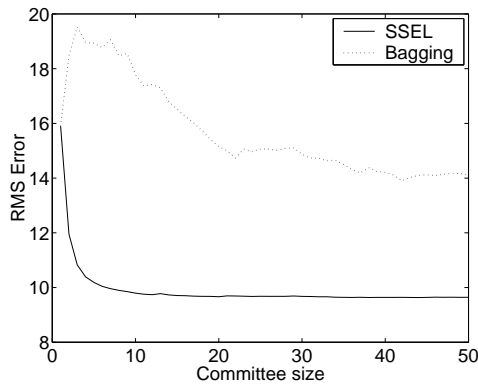


Figure 2.20: SSEL error with 20 records, Boston data

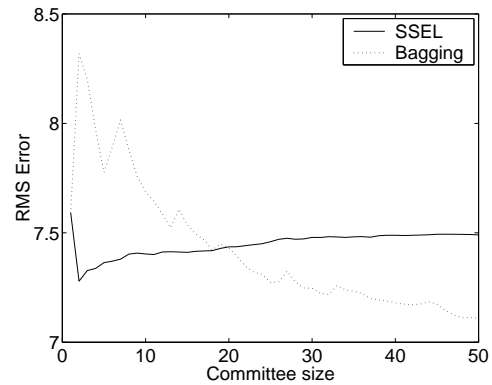


Figure 2.21: SSEL error with 50 records, Boston data

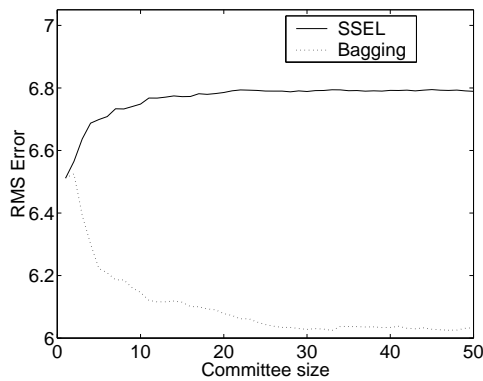


Figure 2.22: SSEL error with 100 records, Boston data

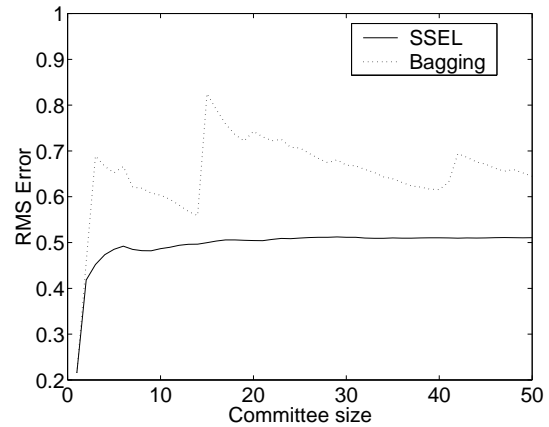


Figure 2.23: SSEL error, meat data

2.22 shows that with a large labelled set, supervised bagging always outperforms semi-supervised, as to be expected. Furthermore, the semi-supervised errors rise (approximately) monotonically, similar to the errors corresponding to larger committee sizes in Figure 2.21.

### 2.11.5 Committee effectiveness

As we discussed in Section 2.5.3, ensembles are used because their errors tend to be lower than the mean of the errors of their members [97]. Table 2.9 shows the committee error and the mean error of individual members, for the three experimental settings with the Boston housing data. As before, the results are averaged over 150 experiments. In all cases, the committee error is lower than the mean member error, as expected. However, the member errors for the semi-supervised case are far lower than for the supervised case, when few training records are available.

The SSEL algorithm had access to a larger set of training data, although the extra records were unlabelled. This goes some way to explain why the SSEL members have lower errors than the supervised bagging members, and hence why SSEL outperforms supervised bagging when the labelled set is very small.

Labelled set size	Bagging		SSEL	
	Committee Error	Member Error	Committee Error	Member Error
20	16.27	34.78	9.88	11.98
50	6.96	10.45	7.10	8.51
100	5.73	7.15	6.29	7.61

Table 2.9: Committee effectiveness — Boston data. The table compares the error of a committee with the average error of the members of that committee.

An alternative explanation for the superior performance of the SSEL algorithm is ridge regression, a form of *shrinkage*. The idea is to shrink the parameters of a regression model towards zero, in order to reduce the variance of the model, and hence improve the generalisation performance. The risk with shrinkage is that it increases the bias of the model, which can lead to larger errors. Given a finite sample of noisy data, ridge regression can lead to better generalisation if a suitable level of shrinkage is chosen. This choice is usually made using cross-validation, e.g. [105, p. 107]. With our small data sets, such cross-validation is difficult, but randomly re-labelling the unlabelled records may have a similar effect, as we now discuss.

Consider a simple linear regression model:  $y = ax + b$ , used to generate some samples of  $x$  and  $y$ . If we estimate the parameters  $a$  and  $b$  from such a sample, then  $a$  estimates the correlation between  $x$  and  $y$ . If we randomly permute the sampled  $y$  values, then we destroy any correlation that the sample may have indicated. If there is effectively no correlation, then our estimate of  $a$  will tend towards zero, and we have a constant model in the form  $y = b$ , with zero variance (but a high bias). Thus if our initial regression model overfits the sample, then subsequent SSEL models will be shrunk, leading to the improved performance we witness.

Noise injection usually refers to the addition of noise to the input values of a neural network, which has a similar effect to weight decay. As with ridge regression, the aim is to reduce the variance of the model, to avoid overfitting. Skurichina [119, p. 46] shows that noise injection is a form of ridge regression for linear discriminant classifiers. In our work with SSEL, we are adding noise to the *labels* of the records, rather than the input features, but with the same result: a reduction in the variance of the model, and an increase in the accuracy.

	Labelled set size							
Unlabelled set size	10	20	30	40	50	60	70	80
0	98.63	14.83	10.36	7.495	7.032	6.172	6.193	6.165
20	21.84 <sup>+</sup>	10.76 <sup>+</sup>	8.458 <sup>+</sup>	7.415	6.849	6.624 <sup>-</sup>	6.434	6.200
40	20.85 <sup>+</sup>	8.857 <sup>+</sup>	8.375 <sup>+</sup>	7.630	7.051	6.846 <sup>-</sup>	6.736 <sup>-</sup>	6.294
60	11.50 <sup>+</sup>	9.401 <sup>+</sup>	8.477 <sup>+</sup>	7.696	7.355	7.148 <sup>-</sup>	6.756 <sup>-</sup>	6.657 <sup>-</sup>
80	12.84 <sup>+</sup>	9.610 <sup>+</sup>	8.953	8.187 <sup>-</sup>	7.342	7.117 <sup>-</sup>	7.228 <sup>-</sup>	6.909 <sup>-</sup>
100	14.46 <sup>+</sup>	9.761 <sup>+</sup>	8.293 <sup>+</sup>	7.855	7.820 <sup>-</sup>	7.342 <sup>-</sup>	7.263 <sup>-</sup>	7.020 <sup>-</sup>
Single model	39.62	15.19	12.68	8.759	7.732	7.544	6.655	6.388
Correlation	-0.721	-0.689	-0.592	0.802	0.889	0.957	0.977	0.969

Table 2.10: Comparison of SSEL errors; single RBF model errors; and correlation between SSEL error and size of unlabelled data set, when using Boston Housing data with varying labelled and unlabelled record set sizes. ‘+’ indicates a significant improvement compared with 0 unlabelled records; ‘-’ indicates a significant worsening compared with 0 unlabelled records; both according to a one-tailed  $t$ -test with  $p < 0.05$ .

### 2.11.6 Varying the amount of unlabelled data

The results shown so far have used 80 unlabelled records, along with various sizes of labelled record sets. We now consider the effects of varying the number of records in both the labelled set and the unlabelled set. We take the 506 records of the Boston Housing data set, and randomly select  $n_L$  records for the labelled set. We then select  $n_U$  of the remaining records for the unlabelled set, and a further 100 records for the independent test set. In each case, we used semi-supervised ensemble learning (SSEL) to build a committee of 50 RBF networks, each with  $\lfloor 2\sqrt{n_L} \rfloor$  Gaussian nodes, as before.

Table 2.10 shows the resultant SSEL error scores, averaged over 50 runs, each with independent samples from the Boston Housing data set. The first column shows that with a very small labelled set (10 records), the effect of the unlabelled data is dramatic: even the addition of just 20 unlabelled records leads to a very large reduction in error. Subsequent extra unlabelled records continue to reduce the error. The table also shows the results of a  $t$ -test comparing the errors to the case when no unlabelled data is available. The leftmost three columns (with 10–30 labelled records) show significant reductions in error in most cases. The rightmost three columns show significant *increases* in error in most cases.

As we increase the size of the labelled set, moving from left to right across the table, the effect of using 20–30 labelled records is less dramatic, but continues to show that the use of unlabelled data allows more accurate models to be built. With 40 or more labelled records, the addition of unlabelled records has the opposite effect: the errors

start rising as we increase the unlabelled set size. These errors are often significantly worse than when no unlabelled data is used.

The penultimate row shows the corresponding error scores obtained by a single RBF network model, estimated using leave one out cross validation. Compared with supervised bagging (i.e. the top row, when no unlabelled data is available), we can see that a single model is more accurate only when a very small labelled set is used. Given 20 or more records, supervised bagging is consistently better than a single model, reflecting the results shown earlier (Tables 2.4–2.6).

The last row of Table 2.10 shows the correlation between the SSEL error scores and the size of the unlabelled data set, for each size of labelled data set studied. The first three values are negative, reflecting the fact that the errors are reduced as the size of the unlabelled set is increased. The remaining correlation scores are positive, showing that for the larger labelled sets, the error increases as we increase the size of the unlabelled set. All the correlation scores are significant according to a  $t$ -test ( $p < 0.01$ ). Note that for a correlation coefficient  $r$  with  $N$  samples, the test statistic used here, namely

$$t = r \sqrt{\frac{N - 2}{1 - r^2}},$$

is distributed according to Student’s  $t$  distribution [86].

These results shows that semi-supervised learning is effective when given a very small labelled set and a second unlabelled set of any size — at least for samples of the Boston Housing data. The appendix includes results on two further public domain data sets (Section A.3). In one case, the results show the same trends as with the Boston Housing data set; in the other case, the results show that SSEL outperforms supervised bagging, even with relatively large labelled set sizes. Both of these sets of results further indicate the potential usefulness of the SSEL technique. The next section applies this semi-supervised ensemble learning to one of the food data sets.

### 2.11.7 Food data results

We now repeat the above experiments using the meat data sets. Here, we have a labelled set of 16 records, and an unlabelled set of 26 records, rather fewer than were used in the initial experiments. Two features were used, as selected earlier by the “simulated annealing cross model validation” method.

Table 2.11 shows the average errors for the three methods. As in the earlier experiments with small labelled data sets, the semi-supervised approach outperforms supervised bagging, and, according to a sign test, this result is statistically significant ( $p = 8.10 \times 10^{-5}$ ). Unlike the previous experiments, the single RBF network significantly outperforms both semi-supervised bagging ( $p = 9.15 \times 10^{-24}$ ) and supervised

	Single	Bagging	SSEL
Mean error	0.215	0.645	0.511
Standard deviation	0.191	1.419	0.609
Maximum error	1.134	139.077	2.752

Table 2.11: Semi-supervised ensemble results — meat data

bagging ( $p = 4.62 \times 10^{-22}$ ). As before, the maximum error score shows that semi-supervised bagging is more consistent than supervised (with a much lower worst-case error and lower standard deviation score).

Figure 2.23 shows the error for bagging and for semi-supervised bagging as the committee size increases. Again, SSEL outperforms supervised bagging for all but the smallest committees. Unlike the Boston Housing experiments, both SSEL and supervised bagging show an initial increase in error as we increase the committee size, confirming that a single model outperforms both committees in this case.

The shape of this graph is quite different to Figures 2.20 – 2.22. In those experiments, at least one of the two committees showed improvement over the initial single-network model. With a small labelled set, SSEL showed considerable improvement; with a large labelled set, supervised bagging showed an equally marked improvement. With the meat data, both get worse. This could imply that both the labelled set *and* the unlabelled set contain too little information for any committee, supervised or semi-supervised, to perform well.

### 2.11.8 SSEL results summary

Given a small set of labelled data and some further unlabelled data, semi-supervised ensemble learning outperforms bagging for a benchmark regression problem. The results for the meat data set are disappointing. Figures 2.20 – 2.22 show that semi-supervised bagging produces the best results with small committees, whereas supervised bagging requires larger committees before the errors level off.

The earlier EM regression results (Section 2.10) show that if the estimated labels are *not* altered in any way, but are simply used to enlarge the training set, then the resultant models are worse than the semi-supervised approach, with the small labelled data sets. For the meat data set, Figure 2.17 shows an error of 1.48, while Table 2.11 shows SSEL has an error of 0.511. Figure 2.18 shows the EM regression algorithm for a small sample (20 records) from the Boston data. Although the error does decrease, the final model has an error of 12.1, which is larger than the SSEL error of 9.3 (Table 2.4). Thus, it is not just the size of the data set which matters, but the re-labelling itself is critical.

The semi-supervised models are closer in output space compared to the supervised bagging models. Their members have errors that are lower but more highly correlated compared to the supervised member errors. This is presumably because they have been trained with a larger data set. The supervised models show very low correlations, and large inter-model distances, when trained with very small data sets. They also had large errors, suggesting that the low correlations here are caused by almost random models, arising from the very small training sets.

Bagging (whether supervised or semi-supervised) works best when we have a weak learner: we start with poor models and improve them through bagging. If the original learner is (close to) perfect, then bagging will have little effect [19]. Similarly, unlabelled data only provides significant extra information if the labelled data set is small. We might call such data a “weak data set” with respect to a given class of learning algorithms (c.f. a “weak learner” with respect to a given data set [49]).

With the larger data set (e.g. with 40 or more labelled records in the Boston data experiments according to Table 2.10), the semi-supervised approach produces *worse* results, compared to the supervised bagged ensemble. If the labelled set is sufficiently large to represent accurately the underlying distribution, then the random labellings applied to the unlabelled records will be misleading, and will decrease the performance. If the first model built is very accurate, it will give accurate estimates of the labels, and so changing the labels will then produce a very misleading training set. We are, in effect, weakening a strong data set, rather than strengthening a weak one.

## 2.12 The effect of the initial number of features

This chapter has highlighted the difficulties of performing feature selection and regression given a large number of features and a small number of records. Selecting from so many features has proved problematic, owing to the vast search space involved. But even if we use an efficient search technique, how accurate will the resultant models be? In particular, what would be the effect of having fewer features to start with?

Given that we are working with given food data sets of a fixed size, we can best approach these questions using synthetic data. This way, we have complete control over the number of features in the data sets. To investigate this, we:

1. Generate a training data set  $D$  with size  $n \times m$ ;
2. Generate a testing data set  $T$  with size  $t \times m$  from the same distribution as  $D$ ;
3. Use  $D$  to perform feature selection, and produce a regression model  $h$ ; and
4. Measure the performance of  $h$  against test data set  $T$ .

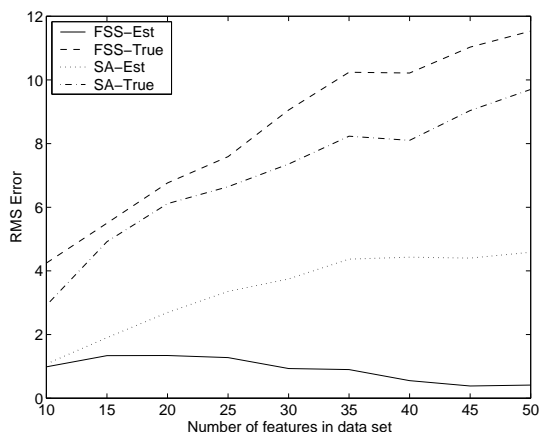


Figure 2.24: Feature selection error, various feature set sizes

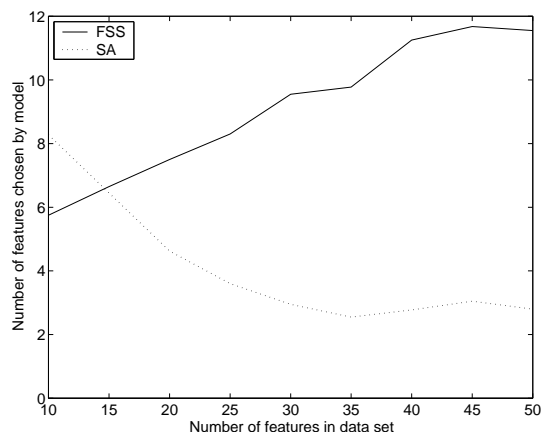


Figure 2.25: Number of features selected, various feature set sizes

If we create  $D$  and  $T$  from the same distribution, then we can test a model built using  $D$  against set  $T$  to give an estimate of the generalization performance. The larger  $T$  gets, the more accurate the error estimate is expected to be. This then gives us a way to compare different feature selection and model-building techniques, while also allowing us to validate the errors which are estimated by the model building process (such as the errors quoted in Tables 2.1–2.2).

To investigate this, we create a training set containing 15 records (similar in size to the sensory data sets) with a much larger test set (500 records) drawn from the same distribution. We vary the number of features in the data set between 5 and 50, with this upper limit being comparable to the sensory data sets. The target of each record is a weighted sum of the normally distributed input values, so the linear regression algorithm used should be able to fit the data closely. We compare two feature selection techniques: the simple forward sequential selection (FSS) method; and simulated annealing (SA) with cross model validation. In each case, we use leave one out cross validation on the training set, to estimate the generalization error, as well as using the test data to get a better estimate of the true generalization error.

If the feature selection techniques work, then both estimates of error should be similar. Hjorth shows that any form of model selection leads to a biased estimate of the error, which will tend to be lower than the true generalization error [66, p.34–38]. Thus, we expect the estimated generalization error to be lower than the true error. The model that we finally select will be the model with the lowest error, so if the estimated error and the true error have the same minimum, we will select the correct model. If the minima do not correspond, then the estimated error will mislead us into choosing the wrong model.

Figures 2.24–2.25 show the results, averaged across 50 experiments. In Figure 2.24,

the bottom line (solid) shows the FSS error estimated from the small training set. As the number of available features increases, the estimated error drops. At the same time, Figure 2.25 shows that the number of features used increases (solid line). However, in Figure 2.24, the top line (dashed) shows that the *true* error, calculated from the large test set, *increases* monotonically. The correlation between the estimated and true errors for FSS is -0.785. Thus, given more features from which to choose, FSS chooses more features and gives a lower estimated error, even though the true error is rising.

Turning to the SA (simulated annealing) with cross model validation results, Figure 2.24 shows that both the estimated error (dotted line) and true error (dash-dot line) rise monotonically. Figure 2.25 shows that the number of features used (dotted line) drops. The correlation between the estimated and true generalization errors for SA is 0.958. Thus, given more features from which to choose, SA chooses fewer to build models, and gives a better estimate of the generalization error, although this is still an under-estimate.

This approach uses synthetic data, and so the results are not directly applicable to the sensory data sets. Nevertheless, the implications are clear: by selecting from a large set of features, the risk of overfitting is greater than if we start with a small set. If we return to the food data set results in Tables 2.1–2.2, this provides some support to our earlier conclusions that the SA cross model validation results are trustworthy, in that they select very few features and give relatively high estimated errors.

## 2.13 Conclusions

In this chapter, we have argued that there are two distinct aspects of feature selection: selecting the number of features to use; and then selecting which features to use. The cross model validation method used here separates these two aspects, solving them each in turn.

The small data sets used in this work fail to provide sufficient information to estimate more than a very small number of parameters in a regression model. Given that the linear models have fewer parameters than the non-linear models, this suggests that we should limit ourselves to linear models. This is vexing, given that we are trying to model nothing less than human taste, which is certainly not linear. For example, suppose a relationship is found between sweetness and preference, showing that people like sweet apples. A linear model suggests that the sweeter the apple, the more it will be liked, without limit. But who would sit down to eat a plate of sugar?<sup>21</sup> The results shown in Section 2.7 show that the non-linear RBF networks often achieve lower generalization errors than the linear regression models, while using slightly fewer

---

<sup>21</sup>Such excessive extrapolations should always be avoided, whatever the nature of the model.

features. By using fewer features but more parameters, the non-linear models perform best. However, this is by no means true in all cases. In general, we run the risk of overfitting the *sample* while underfitting the *population*.

One saving grace of small data sets is that a large number of models can be built and tested rapidly. This allows us to use computationally intensive techniques to search for a good feature set, to an extent that would be impossible if we had many more records in our data set. Of course, with a large data set, we would have sufficient information to ease the feature selection process. Given the currently available data, the results strongly suggest that very few features should be used in model building.

Stochastic searches are computationally expensive, but give good results providing care is taken to avoid overfitting (such as using cross model validation). The semi-supervised “metric-based feature selection” (ADJ, Section 2.8) selects a similar number of features, although with a slightly higher error, compared to the supervised regression and feature selection methods.

The limited volume of labelled data should encourage us to use whatever unlabelled data may be available. Although the information contained therein may be regarded as second rate — in that we are building predictive models, and do not know the values of the labels that we are trying to predict — the novel semi-supervised ensembles and semi-supervised feature selection methods, and the EM regression method all show that this information can still be useful, if the lack of labelled data hinders our attempts to build accurate models.

Feature selection is distinctly non-trivial owing to the combinatorial explosion. In this work, we are further hampered by having a large number of features in the original data set relative to the number of records. With so many features to choose between, no search method can hope to be exhaustive. Instead we must aim to find “quite a good” set of features, with no way of determining if they are in fact optimal, or how much worse than optimal they are. There are two possible solutions to building more accurate models. We could create more records, by using a larger set of food samples. Although the combinatorial explosion would still force us to use heuristic or stochastic feature selection methods, each step of the search could be taken with greater confidence, because the larger data set would allow us to evaluate the intermediate models with greater accuracy. However, the limiting factor here is taste fatigue amongst the preference panellists, restricting how many samples can be used. A more practical solution therefore, is to measure far fewer sensory features, by using expert knowledge at an early stage of the investigation to choose what to measure.

This chapter has concentrated on selecting features from the meat and vegetable data set, because the beverage set  $S_b$  only has eight features to start with. This is because the beverage sensory panel was encouraged to select as few features as possible,

which is unusual.

On the subject of sensory panels and sensory analysis, Stone and Sidel [121, p.209] state that:

It is better to have too many attributes and rely on the analysis to establish which ones were used to differentiate the products.

This is in stark contrast to the results presented here, which show that selecting from an initially large number of features doesn't merely make the analysis more difficult: it makes the final results less accurate.

We must always select which features to use when building models. This can be done using either expert knowledge, or data, or a combination of both. The available data sets are insufficient for this purpose. Therefore, we suggest that the sensory panel should select a small number of features before using them to characterise the food.