Contents

Chapter 9

Algorithm-independent machine learning

9.1 Introduction

In the previous chapters we have seen many learning algorithms and techniques for pattern recognition. When confronting such a range of algorithms, every reader has wondered at one time or another which one is "best." Of course, some algorithms may be preferred because of their lower computational complexity; others may be preferred because they take into account some prior knowledge of the form of the data (e.g., discrete, continuous, unordered list, string, ...). Nevertheless there are classification problems for which such issues are of little or no concern, or we wish to compare algorithms that are equivalent in regard to such issues. In these cases we are left with the question: Are there any reasons to favor one algorithm over another? For instance, given two classifiers that perform equally well on the training set, it is frequently asserted that the *simpler* classifier can be expected to perform better on a test set. But is this version of *Occam's razor* really so evident? Likewise, we frequently prefer or impose *smoothness* on a classifier's decision functions. Do simpler or "smoother" classifiers generalize better, and if so, why? In this chapter we address these and related questions concerning the foundations and philosophical underpinnings of statistical pattern classification. Now that the reader has intuition and experience with individual algorithms, these issues in the theory of learning be better understood.

OCCAM'S RAZOR

In some fields there are strict conservation laws and constraint laws — such as the conservation of energy, charge and momentum in physics, or the second law of thermodynamics, which states that the entropy of an isolated system can never decrease. These hold regardless of the number and configuration of the forces at play. Given the usefulness of such laws, we naturally ask: are there analogous results in pattern recognition, ones that do not depend upon the particular choice of classifier or learning method? Are there any fundamental results that hold regardless of the cleverness of the designer, the number and distribution of the patterns, and the nature of the classification task?

Of course it is very valuable to know that there exists a constraint on classifier

accuracy, the Bayes limit, and it is sometimes useful to compare performance to this theoretical limit. Alas in practice we rarely if ever know the Bayes error rate. Even if we did know this error rate, it would not help us much in designing a classifier; thus the Bayes error is generally of theoretical interest as. What other fundamental principles and properties might be of greater use in designing classifiers?

Before we address such problems, we should clarify the meaning of the title of this chapter. "Algorithm-independent" here refers, first, to those mathematical foundations that do not depend upon the particular classifier or learning algorithm used. Our upcoming discussion of bias and variance is just as valid for methods based on neural networks as for the nearest-neighbor as for model-dependent maximum likelihood. Second, we mean techniques that can be used in conjunction with different learning algorithms, or provide guidance in their use. For example, cross-validation and resampling techniques can be used with any of a large number of training methods. Of course by the very general notion of an algorithm these too are algorithms, technically speaking, but we discuss them in this chapter because of their breadth of applicability and independence from the details of the learning techniques encountered up to here.

In this chapter we shall see, first, that no pattern classification method is inherently superior to any other, or even to random guessing; it is the type of problem, prior distribution and other information that determine which form of classifier is most appropriate. In a particular problem there are differences between classifiers, of course, and thus we show that with certain assumptions we can estimate their accuracy (even before the candidate classifier is fully trained) and compare different classifiers. Second, we shall see methods for integrating component or "expert" classifiers, which themselves might implement any of a number of algorithms.

We shall present the results that are most important for pattern recognition practitioners, occassionally skipping over mathematical details that can be found in the original research referenced in the Bibliography.

9.2 Lack of inherent superiority of any classifier

We now turn to the central question posed above: If we are interested solely in the generalization performance, are there any reasons we should prefer one classifier or learning algorithm to another? If we make no prior assumptions about the nature of the classification task, can we expect any classification method to be superior or inferior overall? Can we even find an algorithm that is overall superior to (or inferior to) random guessing?

9.2.1 No Free Lunch Theorem

As summarized in the *No Free Lunch Theorem*, the answer to these and several related questions is *no*: on the criterion of generalization performance, there are no context or usage independent reasons to favor one learning or classification method over another. The apparent superiority of one algorithm or set of algorithms is due to the nature of the problems investigated and the distribution of data. It is an appreciation of this theorem that allows us, when confronting practical pattern recognition problems, to focus on the aspects that matter most — prior information, data distribution, amount of training data and cost or reward functions. The theorem also justifies a scepticism concerning studies that purport to demonstrate the overall superiority of a particular learning or recognition algorithm.

When comparing algorithms we sometimes focus on generalization error for points *not* in the training set $\mathcal{D}$, rather than the more traditional independent identically distributed or i.i.d. case. We do this for several reasons: First, virtually any powerful algorithm such as the nearest-neighbor algorithm, unpruned decision trees, neural networks with sufficient number of hidden nodes, can learn the training set. Second, for low-noise or low-Bayes error cases, if we use an algorithm powerful enough to learn the training set, then the upper limit of the i.i.d. error decreases as the training set size increases. In short, As such, it is the *off-training set error* — the error on points *not* in the training set — that is a better measure for distinguishing algorithms. Of course, the final performance of a fielded classifier remains the full i.i.d. error.

OFF-TRAINING SET ERROR

For simplicity consider a two-category problem, where the training set $\mathcal{D}$ consists of patterns $\mathbf{x}^i$ and associated category labels $y_i = \pm 1$ for $i = 1, \ldots, n$ generated by the unknown target function to be learned, $F(\mathbf{x})$, where $y_i = F(\mathbf{x}^i)$. In most cases of interest there is a random component in $F(\mathbf{x})$ and thus the same input could lead to different categories, giving non-zero Bayes error. At first we shall assume that the feature set $\mathbf{x}$ is discrete; this simplifies notation and allows the use of summation and probabilities rather than integration and probability densities. The general conclusions hold in the continuous case as well, but the required technical details would cloud our discussion.

Let $\mathcal{H}$ denote the (discrete) set of hypotheses, or possible sets of parameters to be learned. A particular hypothesis $h(\mathbf{x}) \in \mathcal{H}$ could be described by quantized weights in a neural network, or parameters $\boldsymbol{\theta}$ in a functional model, or sets of decisions in a tree, and so on. The prior $P(h)$ is the prior probability that the algorithm will produce hypothesis h after training — this is *not* the probability that h is correct. Next, $P(h|\mathcal{D})$ denotes the probability that the algorithm will yield hypothesis h when trained on the data $\mathcal{D}$. In deterministic learning algorithms such as the nearest-neighbor and decision trees, $P(h|\mathcal{D})$ will be everywhere zero except for a single hypothesis h. For stochastic methods, such as neural networks trained from random initial weights, or stochastic Boltzmann learning, $P(h|\mathcal{D})$ will be a broad distribution. For a general loss function $L(\cdot, \cdot)$ we let $E = L(\cdot, \cdot)$ be the scalar error or cost. While the natural loss function for regression is a sum-squared error, for classification we focus on zero-one loss, and thus the generalization error is the expected value of E.

How shall we judge the generalization quality of a learning algorithm? Since we are not given the target function, the natural measure is the expected value of the error, given $\mathcal{D}$ summed over all possible targets. This scalar value can be expressed as a weighted "inner product" between the distributions $P(h|\mathcal{D})$ and $P(F|\mathcal{D})$, as follows:

$$\mathcal{E}[E|\mathcal{D}] = \sum_{h,F} \sum_{\mathbf{x} \notin \mathcal{D}} P(\mathbf{x})[1 - \delta(F(\mathbf{x}), h(\mathbf{x}))]P(h|\mathcal{D})P(F|\mathcal{D}), \tag{1}$$

where for the moment we assume there is no noise. The familiar Kronecker delta function, $\delta(\cdot, \cdot)$, has value 1 if its two arguments match, and value 0 otherwise. Equation 1 states that the expected error rate, given a fixed training set $\mathcal{D}$, is related to the sum over all possible inputs weighted by their probabilities, $P(\mathbf{x})$, as well as the "alignment" of the learning algorithm, $P(h|\mathcal{D})$, with the actual posterior $P(F|\mathcal{D})$. The important insight provided by this equation is that without prior knowledge concerning $P(F|\mathcal{D})$, we can prove little about any particular learning algorithm $P(h|\mathcal{D})$, including its generalization performance.

The expected off-training set classification error when the true function is $F(\mathbf{x})$ and the learning algorithm is $P_k(h(\mathbf{x})|\mathcal{D})$ is given by

$$\mathcal{E}_k(E|F,n) = \sum_{\mathbf{x}\notin\mathcal{D}} P(\mathbf{x})[1-\delta(F(\mathbf{x}),h(\mathbf{x}))]P_k(h(\mathbf{x})|\mathcal{D}). \tag{2}$$

With this background and the terminology of Eq. 2 we can now turn to a formal statement of the No Free Lunch Theorem.

Theorem 9.1 (No Free Lunch) *For any two learning algorithms $P_1(h|\mathcal{D})$ and $P_2(h|\mathcal{D})$, the following are true, independent of the sampling distribution $P(\mathbf{x})$ and number n of training points:*

1. *Uniformly averaged over all target functions F, $\mathcal{E}_1(E|F,n) - \mathcal{E}_2(E|F,n) = 0$;*
2. *For any fixed training set $\mathcal{D}$, uniformly averaged over F, $\mathcal{E}_1(E|F,\mathcal{D}) - \mathcal{E}_2(E|F,\mathcal{D}) = 0$;*
3. *Uniformly averaged over all priors $P(F)$, $\mathcal{E}_1(E|n) - \mathcal{E}_2(E|n) = 0$;*
4. *For any fixed training set $\mathcal{D}$, uniformly averaged over $P(F)$, $\mathcal{E}_1(E|\mathcal{D}) - \mathcal{E}_2(E|\mathcal{D}) = 0$.*

Part *1* says that uniformly averaged over all target functions the expected error for all learning algorithms is the same, i.e.,

$$\sum_F \sum_{\mathcal{D}} P(\mathcal{D}|F)\left[\mathcal{E}_1(E|F,n) - \mathcal{E}_2(E|F,n)\right] = 0, \tag{3}$$

for any two learning algorithms. In short, no matter how clever we are at choosing a "good" learning algorithm $P_1(h|\mathcal{D})$, and a "bad" algorithm $P_2(h|\mathcal{D})$ (perhaps even random guessing, or a constant output), if all target functions are equally likely, then the "good" algorithm will not outperform the "bad" one. Stated more generally, there are no i and j such that for all $F(\mathbf{x})$, $\mathcal{E}_i(E|F,n) > \mathcal{E}_j(E|F,n)$. Furthermore, no matter what algorithm you use, there is at least one target function for which random guessing is a better algorithm.

Assuming the training set can be learned by all algorithms we consider Part *2* states that even if we know $\mathcal{D}$, then averaged over all target functions no learning algorithm yields an off-training set error error that is superior to any other, i.e.,

$$\sum_F \left[\mathcal{E}_1(E|F,\mathcal{D}) - \mathcal{E}_2(E|F,\mathcal{D})\right] = 0. \tag{4}$$

Parts *3* & *4* concern non-uniform target function distributions, and have related interpretations (Problems 1 – 4). Example 1 provides a very simple illustration.

Example 1: No Free Lunch for binary data

Consider input vectors consisting of three binary features, and a particular target function $F(\mathbf{x})$, as given in the table. Suppose (deterministic) learning algorithm 1 assumes every pattern is in category ω_1 unless trained otherwise, and algorithm 2 assumes every pattern is in ω_2 unless trained otherwise. Thus when trained with $n = 3$ points in $\mathcal{D}$, each algorithm returns a single hypothesis, h_1 and h_2, respectively. In this case the expected errors on the off-training set data are $\mathcal{E}_1(E|F,\mathcal{D}) = 0.4$ and $\mathcal{E}_2(E|F,\mathcal{D}) = 0.6$.

	$\mathbf{x}$	F	h_1	h_2
$\mathcal{D}$	000	1	1	1
	001	-1	-1	-1
	010	1	1	1
	011	-1	1	-1
	100	1	1	-1
	101	-1	1	-1
	110	1	1	-1
	111	1	1	-1

For this target function $F(\mathbf{x})$, clearly algorithm 1 is superior to algorithm 2. But note that the designer does not *know* $F(\mathbf{x})$ — indeed, we assume we have no prior information about $F(\mathbf{x})$. The fact that all targets are equally likely means that $\mathcal{D}$ provides no information about $(\mathbf{x})$. If we wish to compare the algorithms overall, we therefore must average over all such possible target functions consistent with the training data. Part *2* of Theorem 9.1 states that averaged over all possible target functions, there is no difference in off-training set errors between the two algorithms. For each of the 2^5 distinct target functions consistent with the $n = 3$ patterns in $\mathcal{D}$, there is exactly one other target function whose output is inverted for each of the patterns outside the training set, and this ensures that the performances of algorithms 1 and 2 will also be inverted, so that the contributions to the formula in part *2* cancel and thus that part of the Theorem as well as Eq. 4 are obeyed.

Figure 9.1 illustrates a result derivable from Part *1* of Theorem 9.1 (Problem 44). Each of the six squares represents the set of all possible classification problems. Note that this is *not* the standard feature space. If a learning system performs well — higher than average generalization accuracy — over some set of problems, then it must perform worse than average elsewhere, as shown in a). No system can perform well throughout the full set of functions, d); to do so would violate the No Free Lunch Theorem.

In sum, all statements of the form "learning/recognition algorithm 1 is better than algorithm 2" are ultimately statements about the relevant target functions. There is, hence, a "conservation theorem" in generalization: for every possible learning algorithm for binary classification the sum of performance over all possible target functions is exactly zero. Thus we cannot achieve positive performance on some problems without getting an equal and opposite amount of negative performance on other problems. While we may hope that we never have to apply any particular algorithm to certain problems, all we can do is trade performance on problems we do not expect to encounter with those that we do expect to encounter. This, and the other results from the No Free Lunch Theorem, stress that it is the *assumptions* about the learning domains that are relevant. The practical import of the Theorem is that even popular and theoretically grounded algorithms will perform poorly on some problems, ones in which the learning algorithm and the posterior happen not to be "aligned," as governed by Eq. 1. Practitioners must be aware of this possibility, which arises in real-world applications. Expertise limited to a small range of methods, even powerful ones such as neural networks, will not suffice for all classification problems. Experience with a broad range of techniques is the best insurance for solving arbitrary new classification problems.

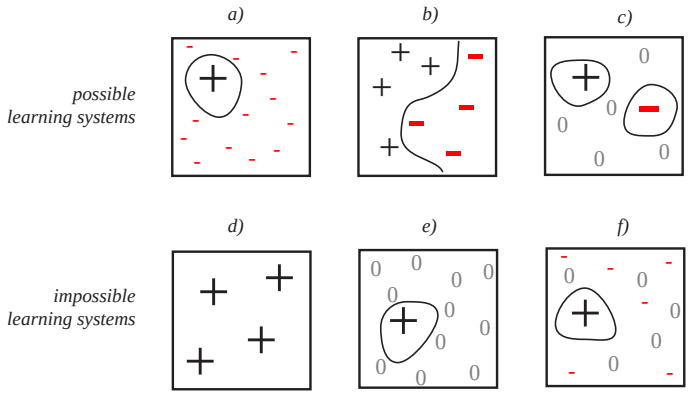


Figure 9.1: The No Free Lunch Theorem shows the generalization performance on the off-training set data that can be achieved (top row), and the performance that is impossible (bottom row). Each square represents all possible classification problems consistent with the training data — this is *not* the familiar feature space. A + indicates that the classification algorithm has generalization higher than average; a − indicates lower than average; a 0 indicates average performance. For instance, a) shows that it is possible for an algorithm to have high accuracy on a small set of problems so long as it has mildly poor performance on all other problems. Likewise, b) shows that it is possible to have excellent performance throughout a large range of problem but this will be balanced by very poor performance on a large range of other problems. It is impossible, however, to have good performance throughout the full range of problems, shown in d). It is also impossible to have higher than average performance on some problems, and average performance everywhere else, shown in e).

9.2.2 Ugly Duckling Theorem

While the No Free Lunch Theorem shows how in the absence of assumptions we should not prefer any learning or classification algorithm over another, an analogous theorem addesses features and patterns. Since we are using discrete representations, we can use logical expressions or "predicates" to describe a pattern, much as in Chap. **??**. Thus a particular pattern might be described by the predicate "x_1 AND x_2," another pattern might be described as "NOT x_2," and so on.

Consider two different patterns and suppose we use a finite number of such predicates to describe them. In the absence of prior information about their distribution, a natural measure the similarity between two patterns would be the number of such predicates they share. The *Ugly Duckling Theorem* states that the number of predicates shared by those two patterns is *constant*, and independent of the patterns themselves (Problem 50). Thus a corollary of the Theorem is that if we judge similarity based on the number of predicates patterns share, then any two distinct patterns are "equally similar."

Theorem 9.2 (Ugly Duckling) *Given that we use a finite set of predicates that enables us to distinguish any two patterns under consideration, the number of predicates shared by any two such patterns is constant and independent of the choice of those patterns. Furthermore, if pattern similarity is based on the total number of predicates*

shared by two patterns, then any two patterns are "equally similar." *

Suppose, for instance, we describe patterns with two apparently "basic" pairs of predicates: BIG/SMALL and RED/NON-RED. This leads to four groupings: 1) BIG *AND* RED, 2) SMALL *AND* RED, 3) BIG *AND* NON-RED and 4) SMALL *AND* NON-RED. But why should we assume these pairs are "basic"? We can easily imagine others predicates, MEISSEN and DELFT, where MEISSEN means BIG *OR* RED *BUT NOT* both, and DELFT means *NOT* MEISSEN. Now we have four other groupings: 1) BIG *AND* MEISSEN, 2) SMALL *AND* MEISSEN, 3) BIG *AND* DELFT and 4) SMALL *AND* DELFT. While these latter four groupings seem unnatural or precluded in the first representation, in fact there is no principled reason to prefer the first representation to the second, since

$$\begin{aligned} \text{RED} &= \text{BIG } OR \text{ MEISSEN } BUT\ NOT\ both \\ \text{MEISSEN} &= \text{BIG } OR \text{ RED } BUT\ NOT\ both. \end{aligned}$$

Consider, for instance, the resentations of a cherry, watermelon and green grape, using the predicates described above:

	RED	NON-RED	BIG	SMALL	MEISSEN	DELFT
cherry	1	0	0	1	1	0
watermelon	0	1	1	0	1	0
grape	0	1	0	1	0	1

This table shows that if we judge objects by the number of predicates they share, all objects share just one predicate are thus "equally similar." In short, there are neither inherently good nor inherently bad features; the appropriate features depend upon the problem. This implies, for instance, that even for a given classifier method such as neural networks, there is no principled reason to favor one choice of features over another. Any such choice amounts to an implied assumption about the properties of the classification problem at hand. For instance, the priors may make RED a more information feature than BIG, but such information applies to the problem, not the classifier. Any choice of classification method or features corresponds to an implicit choice of best or matching prior. Whenever possible, hence, prior information should be incorporated and designers should be prepared to explore a number of methods and features.

9.2.3 Minimum description length (MDL)

It is sometimes claimed that the minimum description length principle provides justification for preferring one type of classifier over another, specifically "simpler" classifiers over "complex" ones. In broad overview, the approach purports to find some irreducible, smallest representation of all members of a category (much like a "signal"); all variation among the individual patterns is then "noise." The principle argues that by simplifying recognizers appropriately, the signal can be retained while the noise is ignored. Because the principle is so often invoked, it is important to understand what properly derives from it, what does not, and how it relates to the No Free Lunch Theorem. To do so, however, we must first understand the notion of algorithmic complexity.

* The Theorem gets its fanciful name from the following counter-intuitive statement: Assuming similarity is based on the number of shared predicates, an ugly duckling is as similar to beautiful swan `A` as beautiful swan `B` is to `A`, given that `A` and `B` differ at all.

Algorithmic complexity

Algorithmic complexity — also known as Kolmogorov complexity, Kolmogorov-Chaitin complexity, descriptional complexity, shortest program length or algorithmic entropy — seeks to quantify an inherent complexity of a binary string. (Below, we shall assume patterns are described by such strings.) Algorithmic complexity can be explained by analogy to communication, the earliest application of information theory (App. **??**). If the sender and receiver agree upon a specification method L, such as an encoding or compression technique, then message x can then be transmitted as y, denoted $L(y) = x$ or $y : L(y) = x$. The cost of transmission of x is the length of the transmitted message y, that is, $|y|$. The least such cost is hence the minimum length of such a message, denoted $\min_{|y|} : L(y) = x$; this minimal length is the entropy of x under the specification or transmission method L.

ABSTRACT COMPUTER

Algorithmic complexity is defined by analogy to entropy, where instead of a specification method L, we consider programs running on an *abstract computer*, i.e., one whose functions (memory, processing, etc.) are described operationally and without regard to hardware implementation. Consider an abstract computer that takes as a program a binary string y and outputs a string x and halts. In such a case we say that y is an abstract encoding or description of x.

A *universal* description should be independent of the specification L (up to some additive constant), so that we can compare the complexities of different binary strings. Such a method would provide a measure of the inherent information content, the amount of data which must be transmitted in the absence of any other prior knowledge. The Kolmogorov complexity of a binary string x, denoted $K(x)$, is defined as the size of the *shortest* program y measured in bits that without additional data, computes the string x and halts. Formally, we write

$$K(x) = \min_{|p|}[U(p) = x], \tag{5}$$

where U represents an abstract universal Turing machine or Turing computer. For our purposes it suffices to state that a Turing machine is "universal" in that it can implement any algorithm and compute any computable function. Kolmogorov complexity is a measure of the incompressibility of x, and is analogous to minimal sufficient statistics, the optimal compression of properties related to a distribution (Chap. **??**).

Consider the following examples of Kolmogorov complexity. Suppose x consists solely of n 1s. This string is actually quite "simple." If we use some fixed number of bits k to specify a general program containing a loop for printing a string of 1s, we need merely $\log_2 n$ more bits to specify the iteration number n, the condition for halting. Thus the Kolmogorov complexity of a string of n 1s is $K(x) = \mathcal{O}(\log_2 n)$. Next consider the transcendental number π, whose infinite sequence of seemingly random binary digits ($11.0010010000111111010110101010001\ldots_2$), actually contains only a few bits of information: the size of the shortest program that can produce any arbitrarily large number of consecutive digits of π. Informally we say the algorithmic complexity of π is a constant; formally we write $K(\pi) = \mathcal{O}(1)$, which means $K(\pi)$ does not grow. Another example is a "truly" random binary string, which cannot be expressed as a shorter string; its algorithmic complexity is within a constant factor of its length. Formally we write $K(x) = \mathcal{O}(|x|)$, which means that $K(x)$ grows as fast as the length of x (Problem 27).

9.2.4 Minimum description length principle

We now turn to a simple, "naive" version of the minimum description length principle and its application to pattern recognition. We imagine that all members of a category share some properties, yet differ in others. The recognizer should seek to learn the common or essential characteristics, while ignoring the accidental or random ones. Kolmogorov complexity provides an objective measure of simplicity, and thus the description of the "essential" characteristics.

Suppose we seek to design a classifier using a training set $\mathcal{D}$. The *minimum description length principle* states we should minimize the sum of the model's algorithmic complexity and the description of the training data with respect to that model, i.e.,

$$K(h, \mathcal{D}) = K(h) + K(\mathcal{D} \text{ using } h). \tag{6}$$

Thus we seek the model h^* that obeys $h^* = \arg\min_h K(h, \mathcal{D})$ (Problem 29). (Variations on the basic minimum description length principle use a *weighted* sum of the terms in Eq. 6.)

A particularly clear application of the minimum description length principle is in the design of decision tree classifiers (Chap. **??**). In this case, a model h specifies the tree and the decisions at the nodes; thus the algorithmic complexity of the model is proportional to the number of nodes. In practice, determining the algorithmic complexity of a classifier depends upon a chosen class of abstract computers, and this means the complexity can be specified only up to an additive constant. The complexity of the data given the model could be expressed in terms of the entropy (in bits) of the data $\mathcal{D}$, the weighted sum of the entropies of the data at the leaf nodes. Thus if the tree is pruned based on an entropy criterion, there is an implicit global cost criterion that is equivalent to minimizing Eq. 6 (Computer exercise 5).

It can be shown theoretically that classifiers designed with a minimum description length principle are guaranteed to converge to the ideal or true model *in the limit of more and more data*. This is surely a very desirable property. However, such derivations cannot prove that the principle leads to superior performance in the *finite* data case; to do so would violate the No Free Lunch Theorems. Moreover, in practice it is often difficult to compute the minimum description length, since we may not be clever enough to find the "best" representation. Given some correspondence between a particular classifier and an abstract computer, it may be simple to determine the length of the string y necessary to create the classifier. Since finding the algorithmic complexity demands we find the *shortest* such string, we must perform a very difficult search through all possible programs that could generate the classifier.

The minimum description length principle can be viewed from a Bayesian perspective as well. Using our current terminology, Bayes formula states

$$P(h|\mathcal{D}) = \frac{P(h)P(\mathcal{D}|h)}{P(\mathcal{D})} \tag{7}$$

for discrete hypotheses and data. The optimal hypothesis h^* is the one yielding the highest posterior probability, i.e.,

$$\begin{aligned} h^* &= \arg\max_h [P(h)P(\mathcal{D}|h)] \\ &= \arg\max_h [\log_2 P(h) + \log_2 P(\mathcal{D}|h)], \end{aligned} \tag{8}$$

much as we saw in Chap. **??**. We note that a string x can be communicated or represented at a cost bounded below by $-\log_2 P(x)$, as stated in Shannon's optimal coding theorem. Shannon's theorem thus provides a link between the MDL (Eq. 6) and the Bayes approaches (Eq. 8). The minimum description length principle states that simple models are to be preferred, and thus amounts to a bias toward "simplicity." It is often easier in practice to specify such a prior in terms of a description length than it is using functions of distributions (Problem 32). We shall revisit the issue of the tradeoff between simplifying the model and fitting the data in the bias-variance dilemma in Sec. 9.3 below.

It is found empirically that classifiers designed using the minimum description length principle work well in many problems. As mentioned, the principle is effectively a method for biasing priors over models toward "simple" models. The reasons for the empirical success of the principle are not trivial, as we shall see in Sect. **??**. One of the greatest benefits of the principle is that it provides a computationally clear approach to balancing model complexity and the fit of the data. In somewhat more heuristic methods, such as pruning neural networks, it is difficult to compare the algorithmic complexity of the network (e.g., number of units or weights) with the entropy of the unexplained data.

9.2.5 Overfitting avoidance, bias and Occam's razor

Throughout our discussions of pattern classifiers, we have mentioned the need to avoid overfitting by means of regularization, pruning, inclusion of penalty terms, minimizing a description length, and so on. The No Free Lunch results above throw these points into question. If there are no problem-independent reasons to prefer one algorithm over another, why is overfitting avoidance nearly universally advocated? For a given training error, why do we generally advocate simple classifiers with fewer features and parameters?

In fact, techniques for avoiding overfitting or minimizing description length are not inherently beneficial; instead, such techniques amount to a preference, or "bias," over the forms or parameters of classifiers. They are only beneficial if they happen to address problems for which they work. It is the match of the learning algorithm to the *problem* — not the imposition of overfitting avoidance — that determines the empirical success. There are problems for which overfitting avoidance actually leads to worse performance (Computer exercise 4). The effects of overfitting avoidance depend upon the choice of representation too. If the feature space is mapped to a new, formally equivalent one, overfitting avoidance has different effects.

In light of the negative results from the No Free Lunch theorems, we might probe more deeply into the frequent "successes" of the minimum description length principle and the more general philosophical principle of Occam's razor. In its original form, Occam's razor stated merely that explanations should not be multiplied beyond necessity, but it has come to be interpreted in pattern recognition as counselling that one should not use classifiers that are more complicated than are necessary, where "necessary" is determined by the quality of fit to the training data. Given the respective requisite assumptions, the No Free Lunch theorem proves that there is no benefit in "simple" classifiers (or "complex" ones, for that matter) — simple classifiers claim neither unique nor universal validity.

The frequent "successes"of Occam's razor imply that the classes of problems addressed so far have certain properties. What might be the reason we explore problems that favor simpler classifiers? A reasonable hypothesis is that through evolution, we

have had strong selection pressure on our pattern recognition apparatuses to be computationally simple — require fewer neurons, less time, and so forth — and in general such classifiers tend to be "simple." We are more likely to ignore problems for which Occam's razor does not hold. Analogously, researchers naturally develop simple algorithms before more complex ones, for instance the progression from the Perceptron, to multilayer neural networks, to networks with pruning, to networks with topology learning, to hybrid neural net/rule-based methods, and so on — each more complex than its predecessor. Each method is found to work on some problems, but not ones that are "too complex." For instance the basic Perceptron is inadequate for optical character recognition; a simple three-layer neural network is inadequate for speaker-independent speech recognition. Hence our design methodology itself imposes a bias toward "simple" classifiers; we generally stop searching for a design when the classifier is "good enough." This principle of *satisficing* — creating an adequate though possibly non-optimal solution — underlies much of practical pattern recognition as well as human cognition. SATISFICING

Another "justification" for Occam's razor derives from a property we might strongly desire or expect in a learning algorithm. If we assume that adding more training data does not, on average, degrade the generalization accuracy of a classifier, then a version of Occam's razor can, in fact, be derived (Problem 48). Note, however, that such a desired property amounts to a non-uniform prior over learning algorithms — while this property is surely desirable, it is a premise and cannot be "proven."

Finally, the No Free Lunch theorem implies that we cannot use training data to create a scheme by which we can with some assurance distinguish new problems for which you generalize well from those for which you generalize poorly (Problem 47).

9.3 Bias and variance

Given that there is no general best classifier unless $p(F)$ is restricted, practitioners must be prepared to explore a number of methods or models when solving a classification problem. Below we will define two ways to measure the "match" or "alignment" of the learning algorithm to the classification problem: the bias and the variance. The bias measures the accuracy or quality of the match: high bias implies a poor match. The variance measures the precision or specificity of the match: a high variance implies a weak match. Designers can adjust the bias and variance of classifiers, but the important bias-variance relation shows that the two terms are not independent; in fact, they obey a strict "conservation law."

9.3.1 Bias and variance for regression

Bias and variance are most easily understood in the context of regression or curve fitting. Suppose there is a true (but unknown) function $F(\mathbf{x})$ with continuous valued output with noise, and we seek to estimate it based on n samples in a set $\mathcal{D}$ generated by, or selected randomly from, $F(\mathbf{x})$. The regression function estimated is denoted $g(\mathbf{x}; \mathcal{D})$ and we are interested in the dependence of this approximation on the training set $\mathcal{D}$. Due to random variations in data selection, for some data sets of finite size this approximation will be excellent while for other data sets of the same size the approximation will be poor. The natural measure of the effectiveness of the estimator can be expressed as its mean square deviation from the desired optimal. Thus we average over all training sets $\mathcal{D}$ of fixed size n and find (Problem 6)

$$\begin{aligned} & \mathcal{E}_{\mathcal{D}}\left[(g(\mathbf{x};\ \mathcal{D}) - F(\mathbf{x}))^2\right] \\ = & \underbrace{(\mathcal{E}_{\mathcal{D}}[g(\mathbf{x};\ \mathcal{D}) - F(\mathbf{x})])^2}_{bias^2} + \underbrace{\mathcal{E}_{\mathcal{D}}\left[(g(\mathbf{x};\ \mathcal{D}) - \mathcal{E}_{\mathcal{D}}[g(\mathbf{x};\ \mathcal{D})])^2\right]}_{variance}. \end{aligned} \tag{9}$$

BIAS

VARIANCE

The first term on the right hand side is the *bias* (squared) — the difference between the expected value and the true (but generally unknown) value — while the second term is the *variance*. Thus a low bias means on average we accurately estimate F from $\mathcal{D}$. Further, a low variance means the estimate of F does not change much as we vary the training set. Even if an estimator is unbiased (i.e., the $bias = 0$ and its expected value is equal to the true value), there can be a large mean square error arising from a large variance term. Equation 9 shows that there is a trade-off between bias and variance; increasing bias reduces the variance, and vice versa. Different classes of regression functions $g(\mathbf{x};\ \mathcal{D})$ — linear, quadratic, sum of Gaussians, etc. — will have different overall errors; nevertheless, Eq. 9 will be obeyed.

Suppose the true, target function $F(x)$ is a cubic polynomial of one variable, with noise, as illustrated in Fig. 9.2. We seek to estimate this function based on a sampled training set $\mathcal{D}$. The left column, a), shows a very poor "estimate" $g(x)$ — a fixed linear function, *independent* of the training data. For different training sets sampled from $F(x)$ with noise, $g(x)$ is unchanged. The histogram of this mean-square error $E \equiv \mathcal{E}_{\mathcal{D}}[\cdot]$, shown at the bottom, reveals a spike at a fairly high error; the variance of the constant model $g(x)$ is zero. Because the estimate is so poor, it has a high bias.

The model in column b) is also fixed, but happens to be a better estimate of $F(x)$. It too has zero variance, but a lower bias than the poor model in a). Presumably the designer imposed some prior knowledge about $F(x)$ in order to get this improved estimate.

The model in column c) is a cubic with trainable coefficients; it would learn $F(x)$ exactly if $\mathcal{D}$ contained infinitely many training points. Notice the fit found for every training set is quite good. Thus the bias is low, as shown in the histogram at the bottom. The model in d) is linear in x, but its slope and intercept are determined from the training data. As such, the model in d) has a lower bias than the models in a) and b).

In sum, for a given target function $F(x)$, if a model has many parameters (generally low bias), it will fit the data well but yield high variance. Conversely, if the model has few parameters (generally high bias), it may not fit the data particularly well, but this fit will not change much as for different data sets (low variance). The best way to get low bias and low variance is the have prior information about the target function. We can virtually never get zero bias and zero variance; to do so would mean there is only one learning problem to be solved, in which case the answer is already known. Furthermore, a large amount of training data will yield improved performance so long as the model is sufficiently general to represent the target function. These considerations of bias and variance helpt to clarify our usual desire to have as much accurate prior information about the form of the solution, and as large a training set as feasible; the match of the algorithm to the problem is crucial.

9.3.2 Bias and variance for classification

While the bias-variance dilemma is simplest to understand in the case of regression, we are most interested in its relevance to classification; here there are a few complications.

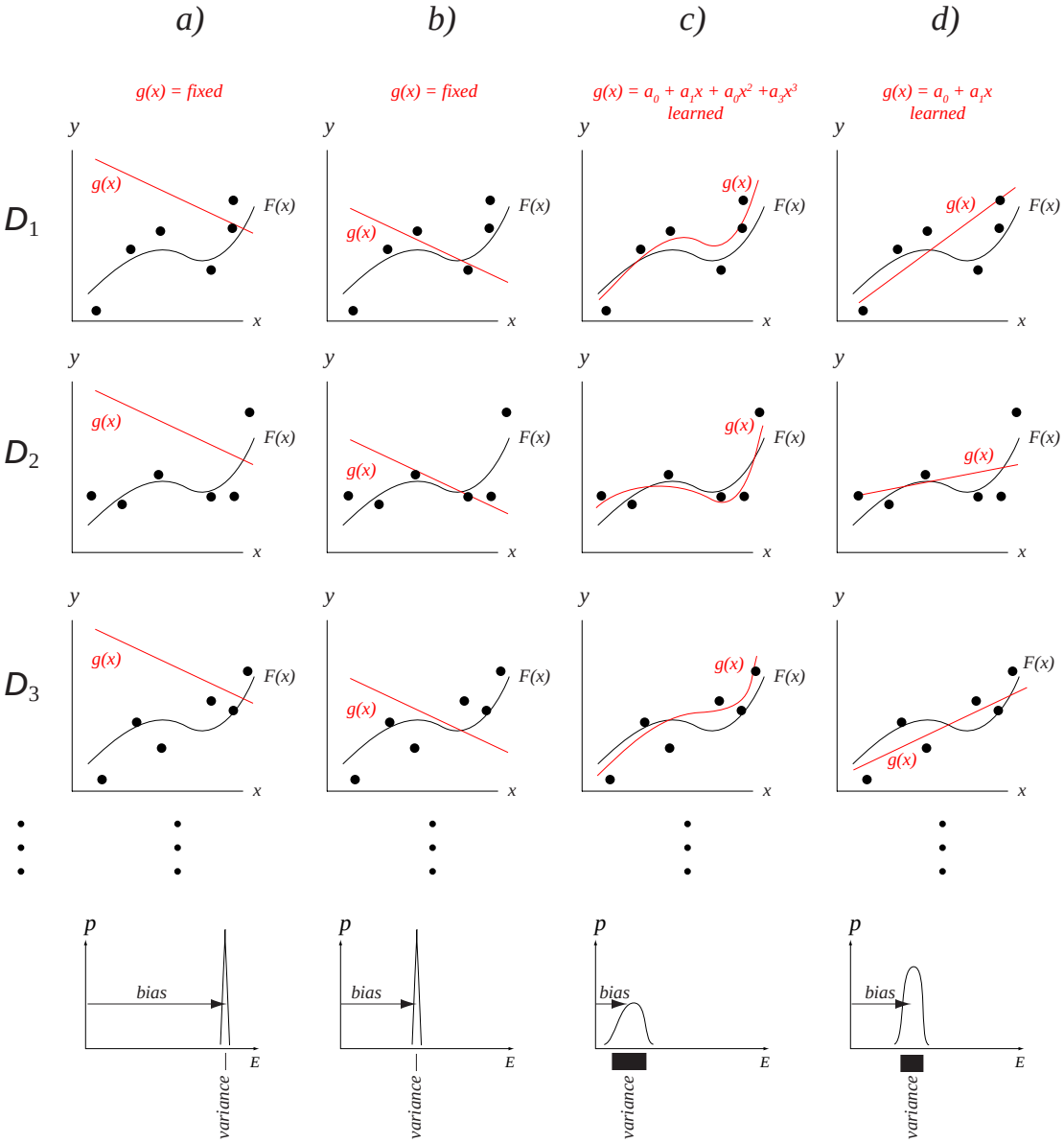


Figure 9.2: The bias-variance tradeoff can be illustrated in the domain of regression. Each column represents a different model, each row a different set of $n = 6$ training points, $\mathcal{D}_i$, randomly sampled from the true function $F(x)$ with noise. Histograms of the mean-squared error of $E \equiv \mathcal{E}_{\mathcal{D}}[(g(x) - F(x))^2]$ of Eq. 9 are shown at the bottom. Column a) shows a very poor model: a linear $g(x)$ whose parameters are held fixed, *independent* of the training data. This model has high bias and zero variance. Column b) shows a somewhat better model, though it too is held fixed, independent of the training data. It has a lower bias than in a) and the same zero variance. Column c) shows a cubic model, where the parameters are trained to best fit the training samples in a mean-squared error sense. This model has low bias, and a moderate variance. Column d) shows a linear model that is adjusted to fit each training set; this model has intermediate bias and variance. If these models were instead trained with a very large number $n \to \infty$ of points, the bias in c) would approach a small value (which depends upon the noise), while the bias in d) would not; the variance of all models would approach zero.

In a two-category classification problem we let the target (discriminant) function have value 0 or +1, i.e.,

$$F(\mathbf{x}) = \Pr[y = 1|\mathbf{x}] = 1 - \Pr[y = 0|\mathbf{x}]. \tag{10}$$

On first consideration, the mean-squared error we saw for *regression* (Eq. 9) does not appear to be the proper one for *classification*. After all, even if the mean-square error fit is poor, we can have accurate classification, possibly even the lowest (Bayes) error. This is because the decision rule, under a zero-one loss, selects the higher posterior $P(\omega_i|\mathbf{x})$, regardless the *amount* by which it is higher. Nevertheless by considering the expected value of y, we can recast classification into the framework of regression we saw before. To do so, we consider a discriminant function

$$y = F(\mathbf{x}) + \epsilon, \tag{11}$$

where ϵ is a zero-mean, random variable, for clarity here assumed to be a centered binomial distribution with variance $\mathrm{Var}[\epsilon|\mathbf{x}] = F(\mathbf{x})(1 - F(\mathbf{x}))$. The target function can thus be expressed as

$$F(\mathbf{x}) = \mathcal{E}[y|\mathbf{x}], \tag{12}$$

and now the goal is to find an estimate $g(\mathbf{x};\ \mathcal{D})$ which minimizes a mean-squared error, such as in Eq. 9:

$$\mathcal{E}_{\mathcal{D}}[(g(\mathbf{x};\ \mathcal{D}) - y)^2]. \tag{13}$$

In this way the regression methods of Sect. 9.3.1 can yield an estimate $g(\mathbf{x};\ \mathcal{D})$ used for classification.

For simplicity we assume equal priors, $P(\omega_1) = P(\omega_2) = 0.5$, and thus the Bayes discriminant y_B has threshold 1/2 and the Bayes decision boundary is the set of points for which $F(\mathbf{x}) = 1/2$. For a given training set $\mathcal{D}$, if the classification error rate, $\Pr[g(\mathbf{x};\ \mathcal{D}) = y]$, averaged over predictions at $\mathbf{x}$, agrees with the Bayes discriminant,

$$\Pr[g(\mathbf{x};\ \mathcal{D}) = y] = \Pr[y_B(\mathbf{x}) \neq y] = \min[F(\mathbf{x}), 1 - F(\mathbf{x})], \tag{14}$$

then indeed we have the lowest error. If not, then the prediction yields an increased error

$$\begin{aligned} \Pr[g(\mathbf{x};\ \mathcal{D})] &= \max[F(\mathbf{x}), 1 - F(\mathbf{x})] \\ &= |2F(\mathbf{x}) - 1| + \Pr[y_B(\mathbf{x}) = y]. \end{aligned} \tag{15}$$

We average over all data sets of size n and find

$$\Pr[g(\mathbf{x};\ \mathcal{D}) \neq y] = |2F(\mathbf{x}) - 1|\Pr[g(\mathbf{x};\ \mathcal{D}) \neq y_B] + \Pr[y_B \neq y]. \tag{16}$$

Equation 16 shows that classification error rate is linearly proportional to $\Pr[g(\mathbf{x};\ \mathcal{D}) \neq y_B]$, which can be considered a *boundary error* in that it represents the mis-estimation of the optimal (Bayes) boundary (Problem 36). This boundary error is analogous to the bias $[\mathcal{E}_{\mathcal{D}}[g(\mathbf{x};\ \mathcal{D}) - F(\mathbf{x})]]^2$ in Eq. 9.

BOUNDARY ERROR

Because of random variations in training sets, the boundary error will depend upon $p(g(\mathbf{x};\ \mathcal{D}))$, the probability density of obtaining a particular estimate of the discriminant given $\mathcal{D}$. This error is merely the area of the tail of $p(g(\mathbf{x};\ \mathcal{D}))$ on

the opposite side of the Bayes discriminant value 1/2, much as we saw in Chap. **??**, Fig. **??**:

$$\Pr[g(\mathbf{x};\ \mathcal{D}) \neq y_B] = \begin{cases} \int_{1/2}^{\infty} p(g(\mathbf{x};\ \mathcal{D}))dg & \text{if } F(\mathbf{x}) < 1/2 \\ \\ \int_{-\infty}^{1/2} p(g(\mathbf{x};\ \mathcal{D}))dg & \text{if } F(\mathbf{x}) \geq 1/2. \end{cases} \tag{17}$$

If we make the natural assumption that $p(g(\mathbf{x};\ \mathcal{D}))$ is a Gaussian, we find (Problem 28)

$$\begin{aligned} \Pr[g(\mathbf{x};\ \mathcal{D}) \neq y_B] &= \Phi\left[sgn[F(\mathbf{x}) - 1/2]\right] \frac{\mathcal{E}_{\mathcal{D}}[g(\mathbf{x};\ \mathcal{D})] - 1/2}{\sqrt{\mathrm{Var}[g(\mathbf{x};\ \mathcal{D})]}} \qquad (18) \\ &= \Phi\Big[\underbrace{sgn[F(\mathbf{x}) - 1/2][\mathcal{E}_{\mathcal{D}}[g(\mathbf{x};\ \mathcal{D})] - 1/2]}_{boundary\ bias} \underbrace{\mathrm{Var}[g(\mathbf{x};\ \mathcal{D})]^{-1/2}}_{variance}\Big], \end{aligned}$$

where

$$\Phi[t] = \frac{1}{\sqrt{2\pi}} \int_{t}^{\infty} e^{-1/2u^2} du = 1 - \mathrm{erf}[t] \tag{19}$$

and erf[·] is the familiar error function (App. **??**).

We have expressed this boundary error in terms of a *boundary bias*, in analogy with the simple bias-variance relation in regression (Eq. 9). Equation 18 shows that the effect of the variance term depends strongly and nonlinearly upon the sign of the boundary bias. In regression the estimation error is *additive* in $bias^2$ and *variance*, whereas for classification there is a nonlinear and *multiplicative* interaction. In classification the sign of the boundary *bias* affects the role of variance in the error. For this reason low *variance* is generally important for accurate classification while low boundary *bias* need not be. Or said another way, in classification, variance generally dominates *bias*. In practical terms, this implies we need not be so concerned if our estimation is biased, so long as the variance is kept low. Numerous specific methods of classifier adjustment — pruning neural networks or decision trees, varying the number of free parameters, etc. — affect the bias and variance of a classifier; in Sect. 9.5 we shall discuss some methods applicable to a broad range of classification methods.

BOUNDARY BIAS

As an illustration of (boundary) bias and variance in classifiers, consider a simple two-class problem in which samples are drawn from two-dimensional Gaussian distributions, each parameterized by vectors $p(\mathbf{x}|\omega_i) \sim N(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$, for $i = 1, 2$. Here the true distributions have diagonal covariances, as shown at the top of Fig. 9.3. We have just a few samples from each category and estimate the parameters in three different classes of models by maximum likelihood. Column a) shows the most general Gaussian classifiers; each component distribution can have arbitrary covariance matrix. Column b) shows classifiers where each component Gaussian is constrained to have a diagonal covariance. The right column is the most restrictive: the covariances are equal to the identity matrix, yielding circular Gaussian distributions. Thus the left column corresponds to very low bias, and the right column to high bias.

Each row in Fig. 9.3 represents a different training set, randomly selected from the true distribution (shown at the top), and the resulting classifiers. Notice that most feature points in the high bias cases retain their classification, regardless of the

particular training set (i.e., such models have low variance), whereas the classification of a much larger range of points varies in the low bias case (i.e., there is high variance). While in general a lower bias comes at the expense of higher variance, the relationship is nonlinear and multiplicative.

At the bottom of the figure, three density plots shows how the location of the decision boundary varies across many different training sets. The left-most density plot shows a very broad distribution — high variance. The right-most plot shows a narrow, peaked distribution — low variance. To visualize the bias, imagine taking the spatial average of the decision boundaries obtained by running the learning algorithm on all possible data sets. The average of such boundaries for the left-most algorithm will be equal to the true decision boundary — this algorithm has no bias. The right-most average will be a vertical line, and hence there will be higher error — this algorithm has the highest bias of the three. Histograms of the generalization error are shown along the bottom.

For a given bias, the variance will decrease as n is increased. Naturally, if we had trained using a very large training set ($n \to \infty$), all error histograms become narrower and move to lower values of E. If a model is rich enough to express the true distributions, its error histogram will approach a delta function at $E = E_B$.

We would like low generalization error, of course, and as mentioned for this it is more important to have low variance than low bias. The only way to get the ideal of zero bias and zero variance is to know the true model ahead of time (or be astoundingly lucky and guess it), in which case no learning was needed anyway. Bias and variance can be lowered with large training size n and accurate prior knowledge of the form of $F(\mathbf{x})$. Further, as n grows, more parameters must be added to the model, g, so the data can be fit (reducing bias). For best classification based on a finite training set, it is desirable to match the form of the model to that of the (unknown) true distributions; this usually requires prior knowledge.

9.4 *Resampling for estimating statistics

When we apply some learning algorithm to a new pattern recognition problem with unknown distribution, how can we determine the bias and variance? Figures 9.2 & 9.3 suggest a method using multiple samples, an inspiration for formal "resampling" methods, which we now discuss. Later we shall turn to our ultimate goal: using resampling and related techniques to improve classification (Sect. 9.5).

9.4.1 Jackknife

We begin with an example of how resampling can be used to yield a more informative estimate of a general statistic. Suppose we have a set $\mathcal{D}$ of n data points x_i ($i = 1, \ldots, n$), sampled from a one-dimensional distribution. The mean is, of course,

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i. \tag{20}$$

Likewise the estimate of the accuracy of μ is the standard deviation σ, defined by

$$\sigma^2 = \frac{1}{n(n-1)}\sum_{i=1}^{n}(x_i - \mu)^2. \tag{21}$$

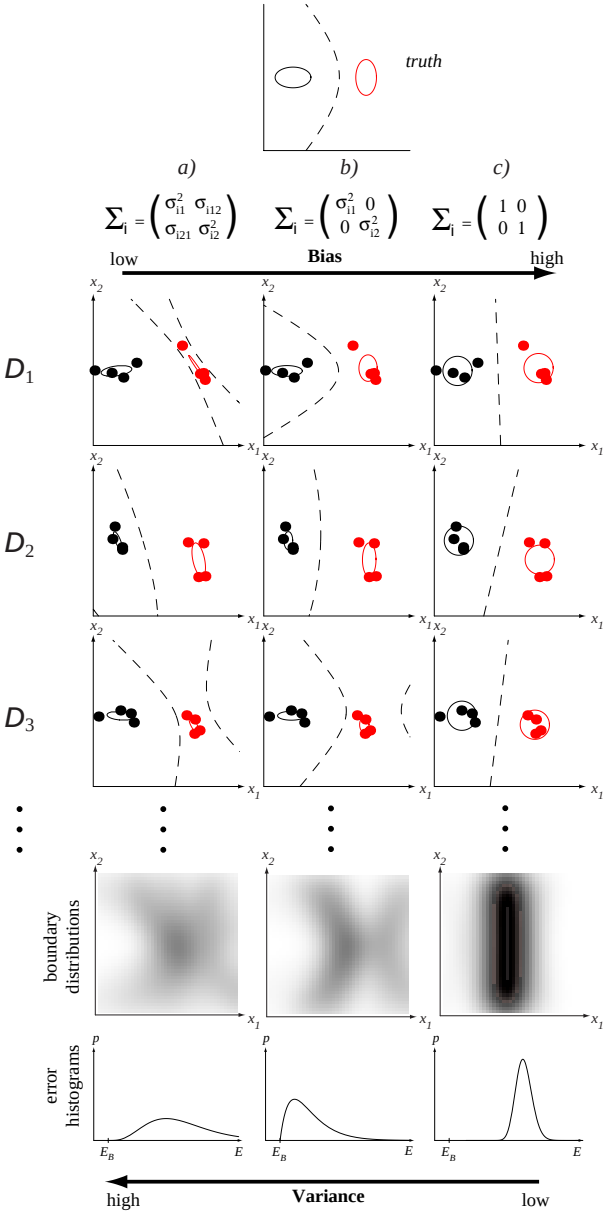


Figure 9.3: The (boundary) bias-variance tradeoff can be illustrated with a two-dimensional Gaussian classification problem. The figure at the top shows the (true) decision boundary of the Bayes classifier. The nine figures in the middle show nine different learned decision boundaries. Each row corresponds to a different training set of $n = 8$ points selected randomly from the true distributions and labelled according to the true decision boundary. The left-most column shows the decision boundaries learning by fitting a Gaussian model with fully general covariance matrices by maximum likelihood. The learned boundaries differ significantly from one data set to the next; this learning algorithm has high variance. The center column shows the decision boundaries resulting from fitting a Gaussian model with diagonal covariances; in this case the decision boundaries vary less from one row to another; this learning algorithm has a lower variance than the one at the left. Finally, the right-most column shows decision boundaries learning by fitting a Gaussian model with unit covariances (i.e., a linear model). Notice that the decision boundaries are nearly identical from one data set to the next; this algorithm has low variance.

MEDIAN

Suppose we were instead interested in the *median*, the point for which half of the distribution is higher, half lower. Although we could determine the median explicitly, there does not seem to be a straightforward way to generalize Eq. 21 to give us a measure of the *error* of our estimate of the median. The same difficulty applies to

MODE

estimating the *mode* (the most frequently represented point in a data set), the 25th percentile, or any of a large number of statistics other than the mean. The jackknife* and bootstrap (Sect. 9.4.2) are two of the most popular and theoretically grounded resampling techniques for extending the above approach to *arbitrary* statistics, of which the mean is just one instance.

In resampling theory, we will frequently use statistics in which one (or more) data point is eliminated from the data; we denote this by means of a special subscript. For

LEAVE ONE OUT

instance, the *leave-one-out* mean is

$$\mu_{(i)} = \frac{1}{n-1}\sum_{j\neq i} x_j = \frac{n\bar{x} - x_i}{n-1}, \tag{22}$$

i.e., the sample average of the data set if the ith point is deleted. Next we define the jackknife estimate of the mean to be

$$\mu_{(\cdot)} = \frac{1}{n}\sum_{i=1}^{n} \mu_{(i)}, \tag{23}$$

that is, the mean of the leave-one-out means. It is simple to prove that the mean and the jackknife estimate of the mean are the same, i.e., $\mu_{(\cdot)} = \mu$ (Problem 7). Likewise, the jackknife estimate of the variance of the estimate obeys

$$\mathrm{Var}[\mu] = \frac{n-1}{n}\sum_{i=1}^{n}(\mu_{(i)} - \mu_{(\cdot)})^2, \tag{24}$$

and, applied to the mean, is equivalent to the familiar variance of Eq. 21 (Problem 17).

The benefit of expressing the variance in the form of Eq. 24 is that we can generalize it to any other estimator $\hat{\theta}$, such as the median or 25th percentile or mode, ... To do so we need to compute the statistic with one data point "left out." Thus we let

$$\hat{\theta}_{(i)} = \hat{\theta}(x_1, x_2, \cdots, x_{i-1}, x_{i+1}, \cdots, x_n) \tag{25}$$

take the place of $\mu_{(i)}$, and let $\hat{\theta}_{(\cdot)}$ take the place of $\mu_{(\cdot)}$ in Eqs. 23 & 24 above.

Jackknife bias estimate

While in Sect. 9.3 we considered the bias in estimating a function $F(\mathbf{x})$, the notion is more general and can be applied to the estimation of any statistic. The bias of an estimator θ is the difference between its true value and its expected value, i.e.,

$$bias = \theta - \mathcal{E}[\theta]. \tag{26}$$

The jackknife method can be used estimate such a bias. The procedure is first to sequentially delete points x_i one at a time from $\mathcal{D}$ and compute the estimate $\hat{\theta}_{(\cdot)}$. The jackknife estimate of the bias is then (Problem 31)

* The jackknife method, which also goes by the name of "leave one out,"was due to Maurice Quenouille. The playful name was chosen by John W. Tukey to capture the impression that the method was handy, and useful in lots of ways.

$$bias_{jack} = (n-1)(\hat{\theta}_{(\cdot)} - \hat{\theta}). \tag{27}$$

We rearrange terms and thus see that the jackknife estimate of θ itself is

$$\tilde{\theta} = \hat{\theta} - bias_{jack} = n\hat{\theta} - (n-1)\hat{\theta}_{(\cdot)}. \tag{28}$$

The benefit of using Eq. 28 is that it is a quadratic function, unbiased for estimating the true bias (Problem 49).

Jackknife variance estimate

Now we seek the jackknife estimate of the variance of an arbitrary statistic θ. First, recall that the standard variance is defined as:

$$\mathrm{Var}[\hat{\theta}] = \mathcal{E}[\hat{\theta}(x_1, x_2, \cdots, x_n) - \mathcal{E}[\hat{\theta}]]^2. \tag{29}$$

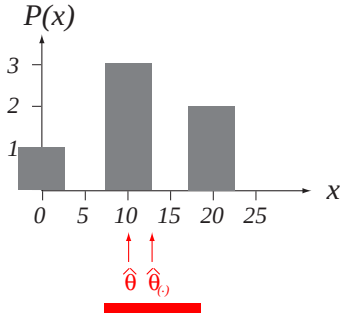
The jackknife estimate of the variance, defined by analogy to Eq. 24, is:

$$\mathrm{Var}_{jack}[\hat{\theta}] = \frac{n-1}{n}\sum_{i=1}^{n}[\hat{\theta}_{(i)} - \hat{\theta}_{(\cdot)}]^2, \tag{30}$$

where as before $\hat{\theta}_{(\cdot)} = \frac{1}{n}\sum_{i=1}^{n}\hat{\theta}_{(i)}$.

Example 2: Jackknife estimate of bias and variance of the mode

As a very simple example, suppose we are interested in the mode of the following $n = 6$ points: $\mathcal{D} = \{0, 10, 10, 10, 20, 20\}$. It is clear from the histogram that the most frequently represented point is $\hat{\theta} = 10$.



A histogram of $n = 6$ points whose mode is $\hat{\theta} = 10$ and jackknife estimate of the mode is $\hat{\theta}_{(\cdot)}$. The square root of the jackknife estimate of the variance is a natural measure of the range of probable values of the mode. This range is indicated by the horizontal red bar.

The jackknife estimate of the mode is

$$\hat{\theta}_{(\cdot)} = \frac{1}{n}\sum_{i=1}^{n}\hat{\theta}_{(i)} = \frac{1}{6}[10 + 15 + 15 + 15 + 10 + 10] = 12.5,$$

where for $i = 2, 3, 4$ we used the fact that the mode of a distribution having two equal peaks is the point midway between those peaks. The fact that $\hat{\theta}_{(\cdot)} > \hat{\theta}$ reveals immediately that the jackknife estimate takes into account more of the full (skewed) distribution than does the standard mode calculation.

The jackknife estimate of the bias of the estimate of the mode is given by Eq. 27:

$$bias_{jack} = (n-1)(\hat{\theta}_{(\cdot)} - \hat{\theta}) = 5(12.5 - 10) = 12.5.$$

Likewise, the jackknife estimate of the variance is given by Eq. 30:

$$\begin{aligned} \mathrm{Var}_{jack}[\hat{\theta}] &= \frac{n-1}{n}\sum_{i=1}^{n}(\hat{\theta}_{(i)} - \hat{\theta}_{(\cdot)})^2 \\ &= \frac{5}{6}[(10-12.5)^2 + 3(15-12.5)^2 + 2(10-12.5)^2] = 31.25. \end{aligned}$$

The square root of this variance, $\sqrt{31.25} = 5.59$, serves as an effective standard deviation. A red bar of twice this width, shown below the figure, reveals that the traditional mode lies within this tolerance to the jackknife estimate of the mode.

The jackknife resampling technique often gives us a more satisfactory estimate of a statistic such as the mode than do traditional methods. While the jackknife technique is especially valuable because it gives us an estimate of the variance of the estimate, the method is of high computational complexity (Problem 18).

Jackknife estimate of classifier accuracy

The application of the jackknife approach to classification is straightforward. We estimate the accuracy of a given algorithm by training the classifier n separate times, each time using a training set $\mathcal{D}$ from which a different single training point has been deleted. Each resulting classifier is tested on the single deleted point and the jackknife estimate of the accuracy is then merely the mean of these leave-one-out accuracies. Likewise, the jackknife estimate of the variance is given by a simple generalization of Eq. 30. Here too the computational complexity may be very high, especially for large n (Problem 24).

9.4.2 Bootstrap

A "bootstrap" data set is one created by randomly selecting n points from the training set $\mathcal{D}$, with replacement. (Since $\mathcal{D}$ itself contains n points, there is nearly always duplication of individual points in a bootstrap data set.) In bootstrap estimation,* the process is independently repeated B times to yield B bootstrap data sets, which are treated as independent sets. The bootstrap estimate of a statistic θ, denoted $\hat{\theta}^{*(\cdot)}$, is merely the mean of the B estimates on the individual bootstrap data sets:

* "Bootstrap" comes from Rudolf Erich Raspe's wonderful stories "The adventures of Baron Munchhausen," in which the hero could pull himself up onto his horse by lifting his own bootstraps. A different but more common usage of the term applies to starting a computer, which must first run a program before it can run other programs.

$$\hat{\theta}^{*(\cdot)} = \frac{1}{B}\sum_{b=1}^{B}\hat{\theta}^{*(b)}, \tag{31}$$

where $\hat{\theta}^{*(b)}$ is the estimate on bootstrap sample b.

Bootstrap bias estimate

The bootstrap estimate of the bias is (Problem 39)

$$bias_{boot} = \frac{1}{B}\sum_{b=1}^{n}\hat{\theta}^{*(b)} - \hat{\theta} = \hat{\theta}^{*(\cdot)} - \hat{\theta}. \tag{32}$$

Computer exercise 51 shows how the bootstrap can be applied to statistics that resist computational analysis, such as the "trimmed mean," in which the mean is calculated for a distribution in which some percentage (e.g., 5%) of the high and the low points in a distribution have been eliminated.

TRIMMED MEAN

Bootstrap variance estimate

The bootstrap estimate of the variance is

$$\mathrm{Var}_{boot}[\theta] = \frac{1}{B}\sum_{b=1}^{B}\left[\hat{\theta}^{*(b)} - \hat{\theta}^{*(\cdot)}\right]^2. \tag{33}$$

If the statistic θ is the mean, then in the limit of $B \to \infty$, the bootstrap estimate of the variance is the traditional variance of the mean (Problem 37).

Generally speaking, the larger the number B of bootstrap samples, the more satisfactory is the estimate of a statistic and its variance. One of the benefits of bootstrap estimation is that B can be adjusted to the computational resources; if powerful computers are available for a long time, then B can be chosen large. In contrast, a jackknife estimate requires exactly n repetitions: fewer repetitions gives a poorer estimate that depends upon the random points chosen; more repetitions merely duplicates information already provided by some of the first n leave-one-out calculations.

Bootstrap estimates of classifier accuracy

There are several ways to generalize the bootstrap method to the problem of estimating the accuracy of a classifier. One of the simplest approaches is to train B classifiers, each with a different bootstrap data set, and test on other bootstrap data sets. The bootstrap estimate of the classifier accuracy is simply the mean of these bootstrap accuracies. In practice, the high computational complexity of bootstrap estimation of classifier accuracy is rarely worth possible improvements in that estimate. We shall discuss a closely related and somewhat useful technique — bagging — in Sect. 9.5.2.

9.5 Resampling for classifier design

The previous section addressed the use of resampling in estimating statistics, including the accuracy of an existing classifier, rather than in the design of classifiers themselves. We now turn to a number of general resampling methods that have proven effective when used in conjunction with any of a wide range of basic classifier methods.

9.5.1 Cross validation

VALIDATION SET

The basic approach in cross validation involves randomly splitting a set of labelled samples $\mathcal{D}$ into two sets: one is used as the traditional training set for adjusting model parameters in the classifier. The other set — the *validation set* — is used to estimate the generalization error. Since our ultimate goal is low generalization error, we train the classifier until we reach a minimum of this validation error, as sketched in Fig. 9.4.

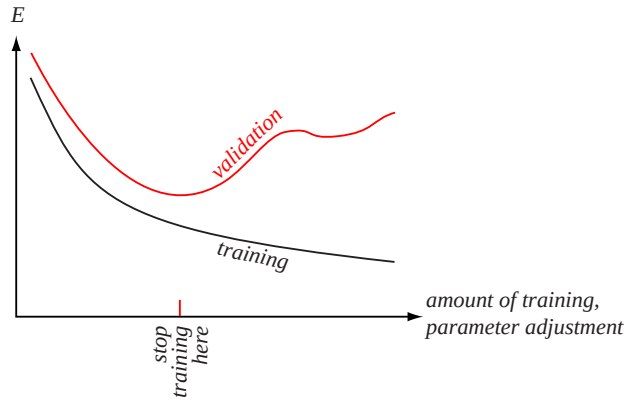


Figure 9.4: In cross validation, the data set $\mathcal{D}$ is split into two parts. The first (e.g., 90% of the patterns) is used as a standard training set for setting free parameters in the classifier model; the other (e.g., 10%) is the validation set and is meant to represent the full generalization task. For most problems, the training error decreases monotonically, during training, as shown in black. Typically, the error on the validation set decreases, but then increases, an indication that the classifier may be overfitting the training data. In cross validation, training or parameter adjustment is stopped at the first minimum of the validation error.

Cross validation can be applied to virtually every classification method, where the specific form of learning or parameter adjustment depends upon the method. For example, in neural networks of a fixed topology (Chap. **??**), the amount of training is the number of epochs or presentations of the training set. Alternatively, the number of hidden units can be set via cross validation. Likewise, the width of the Gaussian window in Parzen windows (Chap. **??**), and an optimal value of k in the k-nearest neighbor classifier (Chap. **??**) can be set by cross validation.

Cross validation is heuristic and need not give improved classifiers in every case. Nevertheless, it is extremely simple and for many real-world problems is found to improve generalization accuracy. There are several heuristics for choosing the portion γ of $\mathcal{D}$ to be used as a validation set ($0 < \gamma < 1$). Nearly always, a smaller portion of the data should be used as validation set ($\gamma < 0.5$) because the validation set is used merely to set a *single* global property of the classifier (i.e., when to stop adjusting parameters) rather than the large number of classifier parameters learned using the training set. If a classifier has a large number of free parameters or degrees of freedom, then a larger portion of $\mathcal{D}$ should be used as a training set, i.e., γ should be reduced. A traditional default is to split the data with $\gamma = 0.1$, which has proven effective in many applications. Finally, when the number of degrees of freedom in the classifier is small compared to the number of training points, the predicted generalization error is relatively insensitive to the choice of γ.

We reiterate that cross validation is a heuristic and need not work on every problem. Indeed, there are problems for which *anti-cross validation* is effective — halting the adjustment of parameters when the validation error is the first local *maximum*. As such, designers must be prepared to explore different γs in any particular problem, and possibly abandon the use of cross validation altogether if performance cannot be improved (Computer exercise 6).

ANTI-CROSS VALIDATION

9.5.2 Bagging

The generic term *arcing* — adaptive reweighting and combining — refers to reusing or selecting data in order to improve classification. While in Sect. 9.5.3 we shall consider the most popular arcing procedure, AdaBoost, here we discuss briefly one of the simplest. Bagging — a name derived from "bootstrap aggregation" — uses multiple versions of a training set, each created by drawing $n' < n$ samples from $\mathcal{D}$ with replacement. Each of these bootstrap data sets is used to train a different *component classifier* and the final classification decision is based on the vote of each component classifier.* Traditionally the component classifiers are of the same general form — i.e., all hidden Markov models, or all neural networks, or all decision trees — merely the final parameter values differ among them due to their different sets of training patterns.

ARCING

COMPONENT CLASSIFIER

A classifier/learning algorithm combination is informally called *unstable* if "small" changes in the training data lead to significantly differerent classifiers and relatively "large" changes in accuracy. As we saw in Chap. **??**, decision tree classifiers trained by a greedy algorithm can be unstable — a slight change in the position of a single training point can lead to a radically different tree. In general, bagging improves recognition for unstable classifiers, since it effectively averages over such discontinuities. There is no convincing theoretical justification that bagging will help all stable classifiers however, a point confirmed in some simulation studies.

INSTABILITY

Bagging is our first encounter with multiclassifier systems, where a final overall classifier is based on the ouputs of a number of component classifiers. The global decision rule in bagging — a simple vote among the component classifiers — is the most elementary method of pooling or integrating the outputs of the component classifiers. We shall consider multiclassifier systems again in Sect. 9.9, with particular attention to forming a single decision rule from the outputs of the component classifiers.

9.5.3 Boosting

The goal of boosting is to improve the accuracy of any given learning algorithm. In boosting we first create a classifier with accuracy on the training set greater than chance, and then add new component classifiers to form an ensemble whose joint decision rule has arbitrarily high accuracy. In such a case we say the classification performance has been "boosted." In overview, the technique trains successive component classifiers with a subset of the training data that is "most informative" given the current set of component classifiers. Classification of a test point $\mathbf{x}$ is based on the outputs of the component classifiers, as we shall see.

For definiteness, consider creating three component classifiers for a two-category problem through boosting. First we randomly select a set of $n_1 < n$ points from the

* In Sect. 9.9 we shall come across other names for component classifiers. For the present purposes, we simply note that these are not classifiers of *component features*, but are instead members in an ensemble of classifiers whose outputs are pooled so as to implement a single classification rule.

full training set $\mathcal{D}$ (without replacement); call this set $\mathcal{D}_1$. Then we train the first classifier, C_1, with $\mathcal{D}_1$. Classifier C_1 need only be a *weak learner*, i.e., have accuracy only slightly better than chance. (Of course, this is the minimum requirement; a weak learner could have high accuracy on the training set but in such a case the benefit of boosting will be small.) Now we seek a second training set, $\mathcal{D}_2$, that is the most informative, given component classifier C_1. Specifically, half of the patterns in $\mathcal{D}_2$ should be correctly classified by C_1, half incorrectly classified by C_1 (Problem 35). Such an informative set $\mathcal{D}_2$ is created as follows: we flip a fair coin. If the coin is heads, we select remaining samples from $\mathcal{D}$ and present them, one by one to C_1 until C_1 misclassifies a pattern. We add this misclassified pattern to $\mathcal{D}_2$. Next we flip the coin again. If heads, we continue through $\mathcal{D}$ to find another pattern misclassified by C_1 and add it to $\mathcal{D}_2$ as just described; if tails we find a pattern which C_1 classifies correctly. We continue until no more patterns can be added in this manner. Thus half of the patterns in $\mathcal{D}_2$ are correctly classified by C_1, half are not. As such $\mathcal{D}_2$ provides information complementary to that represented in C_1. Now we train a second component classifier C_2 with $\mathcal{D}_2$.

WEAK LEARNER

Next we seek a third data set, $\mathcal{D}_3$, which is not well classified by the combined system C_1 and C_2. We randomly select a training pattern from those remaining in $\mathcal{D}$, and classify the selected pattern with C_1 and with C_2. If C_1 and C_2 disagree, we add this pattern to the third training set $\mathcal{D}_3$; otherwise we ignore the pattern. We continue adding informative patterns to $\mathcal{D}_3$ in this way; thus $\mathcal{D}_3$ contains those not well represented by C_1 and C_2. Finally, we train the last component classifier, C_3, with the patterns in $\mathcal{D}_3$.

Now consider the use of the ensemble of three component classifiers for classifying a test pattern $\mathbf{x}$. Classification is based on the votes of the component classifiers. Specifically, if C_1 and C_2 agree on the category label of $\mathbf{x}$, we use that label; if they disagree, then we use the label given by C_3 (Fig. 9.5).

Now we return to a practical detail in boosting: how to choose the number of patterns n_1 to train the first component classifier. We would like to use all patterns in $\mathcal{D}$, and because the final decision is a simple vote among the component classifiers, it is desirable to have roughly equal number of patterns in each (i.e., $n_1 \simeq n_2 \simeq n_3 \simeq n/3$). A reasonable first guess is to set $n_1 \simeq n/3$ and create the three classifiers. If the classification problem is very simple, component classifier C_1 will explain most of the data and thus n_2 (and n_3) will be much less than n_1, and not all of the patterns in the training set $\mathcal{D}$ will be used. Conversely, if the problem is extremely difficult, then C_1 will explain but little of the data, and nearly all the patterns will be informative with respect to C_1; thus n_2 will be unacceptably large. Thus in practice we may need to run the overall boosting procedure a few times, adjusting n_1 in order to use the full training set and, if possible, get roughly equal partitions of the training set. A number of simple heuristics can be used to improve the partitioning of the training set as well (Computer exercise 7).

The above boosting procedure can be applied recursively to the component classifiers themselves, giving a 9-component or even 27-component full classifier. In this way, a very low training error can be achieved, possibly a vanishing training error.

AdaBoost

There are a number of variations on basic boosting. The most popular, AdaBoost — from "adaptive" boosting — allows the designer to continue adding weak learners until some desired low training error has been achieved. In AdaBoost each training

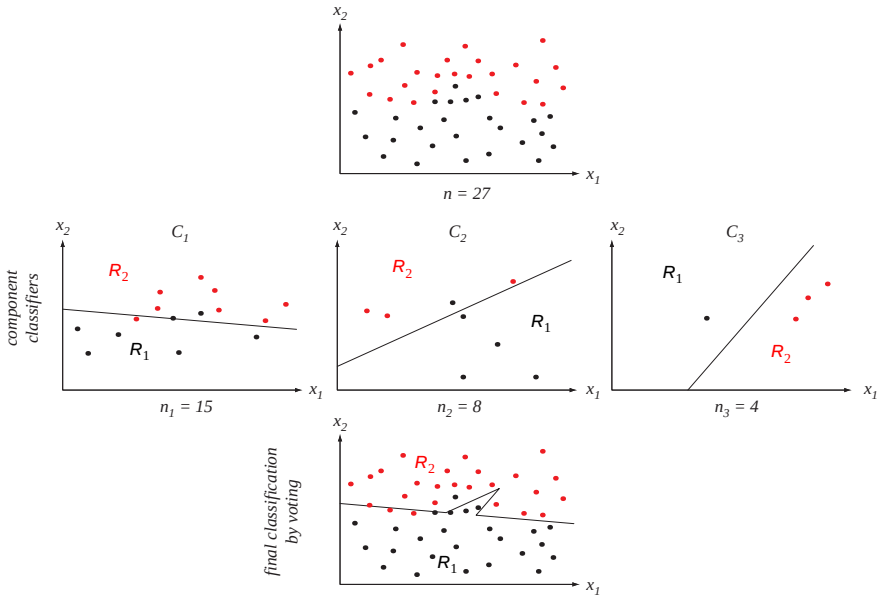


Figure 9.5: A two-dimensional two-category classification task is shown at the top. The middle row shows three component (linear) classifiers C_k trained by LMS algorithm (Chap. **??**), where their training patterns were chosen through the basic boosting procedure. The final classification is given by the voting of the three component classifiers, and yields a nonlinear decision boundary, as shown at the bottom. Given that the component classifiers are weak learners (i.e., each can learn a training set better than chance), then the ensemble classifier will have a lower training error on the full training set $\mathcal{D}$ than does any single component classifier.

pattern receives a weight which determines its probability of being selected for a training set for an individual component classifier. If a pattern is accurately classified, then its chance of being used again in a subsequent component classifier is reduced; conversely, if a pattern is not accurately classified, then its chance of being used again is raised. In this way, AdaBoost "focuses in" on the informative or "difficult" patterns. Specifically, we initialize these weights to to be uniform. On each iteration k, we draw a training set at random according to these weights, and train component classifier C_k on the patterns selected. Next we increase weights of examples misclassified by C_k and decrease weights of examples correctly classified by C_k. Patterns chosen according to this new distribution are used to train the next classifier, C_{k+1}, and the process is iterated.

We let the patterns and their labels in $\mathcal{D}$ be denoted $\mathbf{x}_i$ and y_i, respectively and let $W_k(i)$ be the kth (discrete) distribution over all these training samples. The AdaBoost procedure is then:

Algorithm 1 (AdaBoost)

1 **begin** **initialize** $\mathcal{D} = \{\mathbf{x}_1, y_1, \mathbf{x}_2, y_2, \ldots, \mathbf{x}_n, y_n\}, W_1(i) = 1/n, k_{max}$
2 $k \leftarrow 0$
3 **do** $k \leftarrow k + 1$
4 Train weak learner C_k using $\mathcal{D}$ sampled according to distribution $W_k(i)$
5 $E_k \leftarrow$ Training error of C_k measured on $\mathcal{D}$ using $W_k(i)$

$\alpha_k \leftarrow \frac{1}{2}\ln[(1-E_k)/E_k]$

$W_{k+1}(i) \leftarrow \frac{W_k(i)}{Z_k} \times \begin{cases} e^{-\alpha_k} & \text{if } h_k(\mathbf{x}_i) = y_i \text{ (correctly classified)} \\ e^{\alpha_k} & \text{if } h_k(\mathbf{x}_i) \neq y_i \text{ (incorrectly classified)} \end{cases}$

until $k = k_{max}$

return C_k and α_k for $k = 1$ to k_{max} (ensemble of classifiers with weights)

end

Note especially in line 5 that the error for classifier C_k is determined with respect to the distribution $W_k(i)$ over $\mathcal{D}$ on which it was trained. In line 7, Z_k is simply a normalizing constant computed to insure that $W_k(i)$ represents a true distribution, and $h_k(\mathbf{x}_i)$ is the category label (+1 or -1) given to pattern $\mathbf{x}_i$ by component classifier C_k. Naturally, the loop termination criterion in line 8 can be replaced by the criterion of sufficiently low training error of the ensemble classifier.

The final classification decision of a test point $\mathbf{x}$ is based on a discriminant function that is merely the weighted sums of the outputs given by the component classifiers:

$$g(\mathbf{x}) = \left[\sum_{k=1}^{k_{max}} \alpha_k h_k(\mathbf{x})\right]. \tag{34}$$

The classification decision for this two-category case is then simply $\text{sgn}[g(\mathbf{x})]$.

So long as each component classifier is a weak learner, the total training error of the ensemble can be made arbitrarily low by setting the number of component classifiers, k_{max}, sufficiently high. To see this, notice that the training error for weak learner C_k can be written as $E_k = 1/2 - G_k$ for some positive value G_k. Thus the ensemble training error is (Problem 41):

$$\begin{aligned} E = \prod_{k=1}^{k_{max}} \left[2\sqrt{E_k(1-E_k)}\right] &= \prod_{k=1}^{k_{max}} \sqrt{1-4G_k^2} \\ &\leq \exp\left(-2\sum_{k=1}^{k_{max}} G_k^2\right), \end{aligned} \tag{35}$$

as illustrated in Fig. 9.6. It is sometimes beneficial to increase k_{max} beyond the value needed for zero ensemble training error as this may improve generalization. While in theory a large k_{max} could lead to overfitting, simulation experiments have shown that overfitting rarely occurs, even when k_{max} is extremely large.

At first glance, it appears that boosting violates the No Free Lunch Theorem in that an ensemble classifer seems to always perform better than any single component classifier on the full training set. After all, according to Eq. 35 the training error drops exponentially fast with the number of component classifiers. The Theorem is not violated, however: boosting only improves classification if the component classsifiers performs *better than chance*, but this cannot be guaranteed a priori. If the component classifiers cannot learn the task better than chance, then we do not have a strong match between the problem and model, and should choose an alternate learning algorithm. Moreover, the exponential reduction in error on the training set does not insure reduction of the off-training set error, as we saw in Sect. 9.2.1.

9.5.4 Learning with queries

In the previous sections we assumed there was a set of labelled training patterns $\mathcal{D}$; we employed resampling methods to reuse informative patterns to improve classifica-

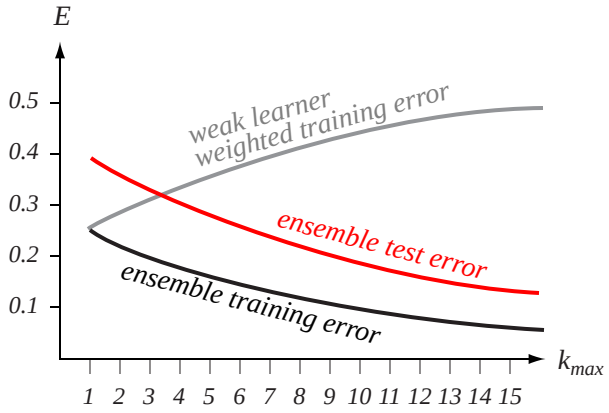


Figure 9.6: AdaBoost applied to a weak learning system can reduce the training error E exponentially as the number of component classifiers, k_{max}, is increased. Because AdaBoost "focusses on" difficult training patterns, the training error of each successive component classifier (measured on its own weighted training set) is generally larger than that of any previous component classifier (shown in gray). Nevertheless, so long as the component classifiers perform better than chance, the weighted ensemble decision of Eq. 34 insures that the training error decreases, as given by Eq. 35. It is often found that the test error decreases in boosted systems as well, as shown in red.

tion. In many applications, however, the patterns are *unlabelled*. We shall return in Chap. **??** to learning when no labels are available but here we assume there exists some possibly costly way of labelling them. Our current challenge in learning with queries hence is to determine which unlabelled patterns would be most informative, if labelled. These are the ones we will present to an *oracle* — a teacher who can label, without error, any pattern. Thus we seek an automatic way of selecting unlabelled patterns to present as a *query* to the oracle; the pattern and its label supplied by the oracle are then used to train the classifier. This approach is called variously learning with queries, active learning or interactive learning.

ORACLE

Suppose, for example, we wanted to design a classifier for handwritten numerals using a very large corpus of unlabelled pixel images as might be scanned from postal letter addresses. We could start the process by choosing at random a small number of patterns, labelling them, and training our classifier in traditional way. Because accurate classification requires a large number of labelled patterns, however, we now must use learning with queries to select those unlabelled patterns from our set to present to a human — the oracle — for labelling. Informally, we would expect the most valuable patterns would be near the final decision boundaries.

We assume some preliminary, weak classifier has been developed with a small set of labelled samples. We now must choose an unlabelled pattern to present as queries to the oracle for labelling. There are two related methods for selecting such an informative pattern. In *confidence based* query selection, we seek the patterns for which the current classifier is least certain. Suppose the classifier computes discriminant functions $g_i(\mathbf{x})$ for the c categories, $i = 1, \ldots, c$. Thus the classifier is not confident about its classification of a point $\mathbf{x}$ if the two largest discriminant functions have nearly the same value. This will occur for patterns near decision boundaries.

CONFIDENCE BASED QUERY SELECTION

The second method, *voting based* or *committee based* query selection, is similar to the previous method but is applicable to multiclassifier systems, that is to say

VOTING BASED QUERY SELECTION

ones comprising several component classifiers. Each unlabelled pattern is presented to each of the k component classifiers; the pattern that yields the greatest disagreement among the k category labels is considered the most informative pattern, and is thus presented as a query to the oracle. Voting based query selection can be used even if the component classifiers do not provide analog discriminant functions, for instance decision trees, rule-based classifiers or simple k-nearest neighbor classifiers. In both these methods, the pattern labelled by the oracle is then used for training the classifier in the traditional way. (We shall return in Sect. **??** to training an ensemble of classifiers.)

Clearly such learning with queries does not directly exploit information about the prior distribution of the patterns. For instance, in most problems the distributions of query patterns lie near the final decision boundaries (where patterns are informative) rather than at the region of highest prior probability (where they are less informative), as illustrated in Fig. 9.7. One benefit of learning with queries is that we need not guess the form of the underlying distribution, but can instead use a non-parametric technique, such as nearest neighbor classification, that allows the boundary to be found accurately.

If there is not a large set of unlabelled samples for queries, we can exploit learing with queries if there is a way to *generate* query patterns. Suppose we have a only small set of labelled handwritten characters. Suppose too we have image processing algorithms for altering these images to present as new, surrogate patterns as queries to the oracle. For instance, the pixel images might be rotated, scaled, sheared, have random noise added or their lines thinned. Further, we might be able to generate new patterns "in between" two others in the set by interpolating or mixing them. With such generated queries the classifier can explore regions of the feature space about which it is least confident (Fig. 9.7).

9.5.5 Arcing, learning with queries, bias and variance

In Chap. **??** and many other places, we have stressed the need for training a classifier on samples drawn from the distribution on which it will be tested. Resampling in general, and learning with queries in particular, seem to violate this recommendation. Why can a classifier trained on a strongly weighted distribution of data be expected to do well — or better! — than one trained on the i.i.d. sample? Why doesn't resampling lead to *worse* performance, to the extent that the resampled distribution differs from the i.i.d. one? Indeed, if we were merely training a model of the true distribution with a highly skewed distribution obtained by learning with queries, the final accuracy might be unacceptably low. But note that in learning with queries this is not what we do.

Consider two interrelated points about resampling methods and altered distributions. The first is that when we use resampling methods, we generally use classification and learning methods that do not model or fit the full category distributions. Thus even if we suspect the prior distributions for two categories might be Gaussians, we might instead use a nonparametric method such as nearest neighbor, radial basis function, or RCE classifiers. Thus in learning with queries we are not fitting parameters in a model, as described in Chap. **??**, but instead are seeking decision boundaries more directly.

The second point is that techniques such as general boosting and AdaBoost effectively broaden that class of implementable functions as the number of component classifiers is increased, as illustrated in Fig. 9.5. While the final classifier might indeed

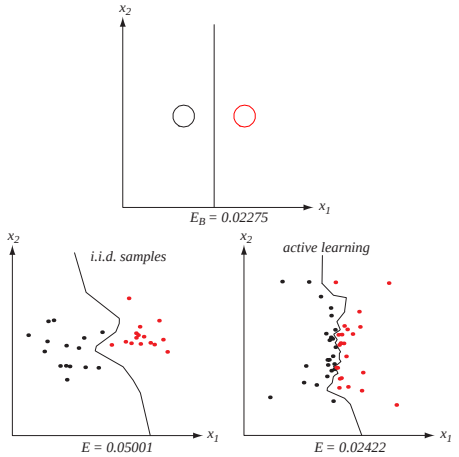


Figure 9.7: Active learning can be used to create classifiers that are more accurate than ones using i.i.d. sampling. The figure at the top shows a two-dimensional problem with two equal circular Gaussian priors; the Bayes decision boundary is a straight line and the Bayes error $E_B = 0.02275$. The bottom figure on the left shows a nearest-neighbor classifier trained with $n = 30$ labelled points sampled i.i.d. from the true distributions. Note that most of these points are far from the decision boundary. The figure at the right illustrates active learning. The first four points were sampled near the extremes of the feature space. Subsequent query points were chosen midway between two points already used by the classifier, one randomly selected from each of the two categories. In this way, successive queries to the oracle "focused in" on the true decision boundary. The final generalization error of this classifier (0.02422) is lower than the one trained using i.i.d. samples (0.05001).

be characterized as parametric, it is in an expanded space of parameters, one larger than that of the first component classifier.

In broad overview, though, cross validation, resampling, boosting and related procedures are heuristic methods for adjusting the class of implementable functions. As such they are indirectly adjusting the bias and variance of an arbitrary classification scheme. Given the importance of matching the classifier to a problem, heuristic resampling methods can be extremely useful to this end.

9.6 Estimating and comparing classifiers

There are at least two reasons for wanting to know the error rate of a classifier on a given problem. One is to see if the clasifier performs well enough to be useful. Another is to compare its performance with a competing design.

One approach to estimating the error rate is to compute it from the assumed parametric model. For example, in the two-class multivariate normal case, one might compute $P(E)$ using the Bhattacharyya or Chernoff bounds (Chap **??**), substituting estimates of the means and the covariance matrix for the unknown parameters. However, there are three problems with this approach. First, such an estimate for $P(e)$ is almost always overly optimistic; characteristics that make the design samples peculiar or unrepresentative will not be revealed. Second, one should always suspect the validity of an assumed parametric model; a performance evaluation based on the

same model can not be believed unless the evaluation is unfavorable. Finally, in more general situations it is very difficult to compute the error rate exactly, even if the probabilistic structure is complete known. Another way is to use the jackknife or bootstrap estimation procedures (Sect. 9.4.1 & 9.4.2). The jackknife, in particular, generally gives excellent estimates, since each of the the n classifiers is quite similar to the classifier being tested (differing soley dut to a single training point.)

A particular benefit of the jackknife is that it permits meaningful comparison between accuracies of two competing classifier designs. Suppose trained classifier C_1 has an accuracy of 80% while C_2 has accuracy of 85%, as estimated by the jackknife procedure. Is C_2 really better than C_1? To answer this, we calculate the jackknife estimate of the variance of the classification accuracies and use traditional hypoththesis testing to see if C_1's apparent superiority is statistically significant (Fig. 9.8).

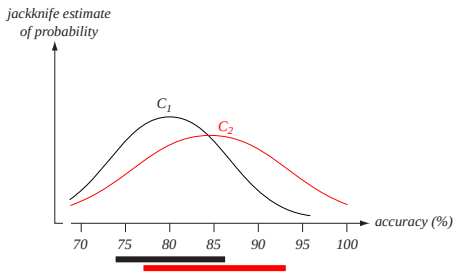


Figure 9.8: Jackknife estimation can be used to compare the accuracies of classifiers. The jackknife estimate of classifiers C_1 and C_2 are 80% and 85%, and full widths (twice the square root of the jackknife estimate of the variances) are 12% and 15%, respectively. In this case, traditional hypothesis testing shows that the difference is not statistically significant.

An empirical approach that avoids these problems is to test the classifier experimentally. In practice, this is frequently done by running the classifier on a set of *test samples*, using the fraction of the samples that are misclassified as an estimate of the error rate. Needless to say, the test samples should be different from the design or test samples, otherwise the estimated error rate will definitely be optimistic — a flaw called "testing on the training data." If the true but unknown error rate of the classifier is p, and if k of the n independent, randomly drawn test samples are misclassified, then k has the binomial distribution

$$P(k) = \binom{n}{k} p^k (1-p)^{n-k}. \tag{36}$$

Thus, the fraction of test samples misclassified is exactly the maximum likelihood estimate for p:

$$\hat{p} = \frac{k}{n}. \tag{37}$$

The properties of this estimate for the parameter p of a binomial distribution are well known. In particular, Figure ?? shows 95 percent confidence intervals as a function of $\hat{p}$ and n. For a given value of $\hat{p}$, the probability is 0.95 that the true value of p lies in the interval between the lower and upper curves for the number n of test samples. These curves show that unless n is fairly large, the maximum likelihood estimate must be interpreted with caution. For example, if no errors are made on 50 test samples, with probability 0.95 the true error rate is between zero and eight percent. The classifier would have to make no errors on more than 250 test samples to be reasonably sure that the true error rate is below two percent.

The need for data to design the classifier and additional data to evaluate it presents the designer with a dilemma. If he or she reserves most of the data for the design, then there will be little confidence in the test. If most of the data is used for the test, then there will be a poor design.

9.6.1 Learning curves

Learning curves (compare classifiers, sort of)

Goal: know whether a method shows promise, before spending a great deal of time training the classifier.

$$E_{test} = a + b/m^{\alpha} \tag{38}$$

and

$$E_{train} = a - c/m^{\beta} \tag{39}$$

text here

$$E_{test} + E_{train} = 2a + \frac{b}{m^{\alpha}} - \frac{c}{m^{\beta}} \tag{40}$$

$$E_{test} + E_{train} = \frac{b}{m^{\alpha}} + \frac{c}{m^{\beta}}. \tag{41}$$

The assumption $b = c$ is not crucial; if this approximation does not hold, the difference $E_{test} - E_{train}$ still forms a straight line on a log-log plot. The sum $s = b + c$ can be found from the intersection, as shown in Fig. **??**

text here

9.6.2 Predicting asymptotic error

Predict the asymptotic error rate

9.7 Model selection

In many applications we have only a vague notion of the model underlying distributions or the appropriate classification method. Thus we need a principled means to select among different candidate *models*, particularly when they are of different structure, for instance comparing a k-nearest neighbor with a three-layer neural net. One straightforward approach is to simply train each classifier using the training data and compare their performance on a test set or instead use jackknife estimation. However, different models have different numbers of free parameters and different biases, and thus it is not always clear which algorithm is to be preferred.

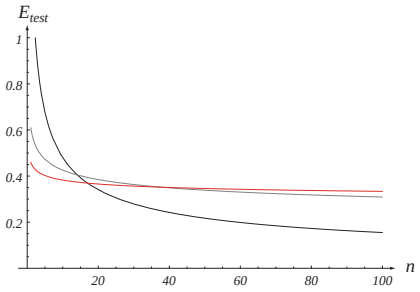


Figure 9.9: The test error for three classifiers show a typical monotonic decrease versus the number of training patterns. Notice that the rank order of the classifiers trained on $n = 5$ points differs from that on $n = 100$ points.

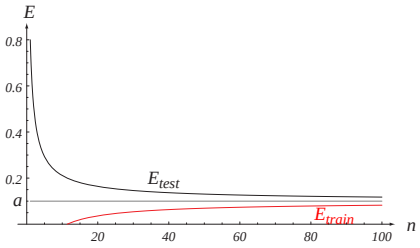


Figure 9.10: Test and training error versus training set size n. Note this is the final error after full training on n points. For instance, at low n, the classifier can learn the category labels of the points perfectly, and thus the error vanishes. In the limit $n \to \infty$, both training and test errors approach the same asymptotic value, a.

9.7.1 Maximum likelihood approach

Maximum likelihood can be applied to *model* selection. In maximum likelihood approaches, assuming equal model priors $P(H_i)$, we should chose the model having the greatest likelihood $P(\mathcal{D}|H_i)$. Bayesian model selection employs an integration over a model's parameter space $\boldsymbol{\theta}$, which can be approximated by a best fit likelihood times an Occam factor, the later "penalizing" the model for having parameter $\hat{\boldsymbol{\theta}}$.

Recall maximum likelihood parameter estimation as discussed in Chap. **??**. Given a model with unknown parameter vector $\boldsymbol{\theta}$, we find the value $\hat{\boldsymbol{\theta}}$ which maximizes the probability of the training data, i.e., $p(\mathcal{D}|\boldsymbol{\theta})$.

Maximum likelihood *model selection* — sometimes called ML-II — is a direct generalization of those maximum likelihood techniques. The goal here is to chose the model that best explains the training data, in a way that will become clear below. As above, we let $h_i \in \mathcal{H}$ represent a candidate hypothesis or model, and $\mathcal{D}$ the training data. The posterior probability of any given model is given by Bayes rule:

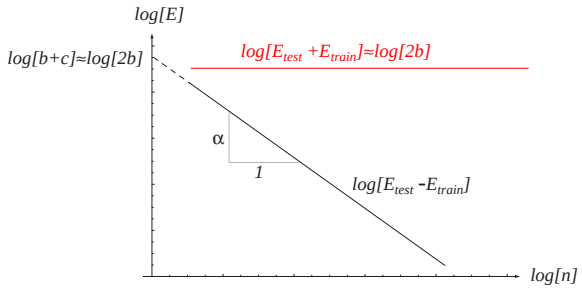


Figure 9.11: Log of test + training error versus training set size

$$P(h_i|\mathcal{D}) = \frac{P(\mathcal{D}|h_i)P(h_i)}{p(\mathcal{D})} \propto P(\mathcal{D}|h_i)P(h_i), \tag{42}$$

where we will rarely need the normalizing factor $p(\mathcal{D})$. The data-dependent term, $P(\mathcal{D}|h_i)$, is the *evidence* for h_i; the second term, $P(h_i)$, is our subjective prior over our space of hypotheses — it rates our confidence in different models even before the data arrives. In practice, the data-dependent term dominates, and hence the priors $P(h_i)$ are often neglected in the computation. In short, we find the maximum likelihood parameters for each of the candidate models, calculate the resulting likelihoods, and select the model with the largest such likelihood. EVIDENCE

There are two important subtleties. The first is that of degeneracy: for any model there may be several values of the parameter $\boldsymbol{\theta}$ that give the maximum likelihood. The second is that different models may have different numbers of parameters or degrees of freedom. While these concerns affect maximum likelihood model selection, they are more naturally dealt with in the Bayesian counterpart, and we thus now turn to that method.

[[example here]]

9.7.2 Bayesian approach

Bayesian model selection requires we use the full information over priors. In particular, we write the evidence for a particular hypothesis as:

$$P(\mathcal{D}|h_i) = \int p(\mathcal{D}|\boldsymbol{\theta}, h_i)p(\boldsymbol{\theta}|\mathcal{D}, h_i)d\boldsymbol{\theta}. \tag{43}$$

It is common for the posterior $P(\boldsymbol{\theta}|\mathcal{D}, h_i)$ to be peaked at $\hat{\boldsymbol{\theta}}$, and thus the evidence integral can often be approximated as:

$$p(\mathcal{D}|h_i) \simeq \underbrace{P(\mathcal{D}|\hat{\boldsymbol{\theta}}, h_i)}_{\text{best fit likelihood}} \underbrace{p(\hat{\boldsymbol{\theta}}|h_i)\Delta\boldsymbol{\theta}}_{\text{Occam factor}} . \tag{44}$$

Before the data arrive, model h_i could account for some broad range of the data, denoted by $\Delta^0\boldsymbol{\theta}$, as shown in Fig. 9.12. The *Occam factor* OCCAM FACTOR

$$\text{Occam factor} = \frac{\Delta\boldsymbol{\theta}}{\Delta^0\boldsymbol{\theta}} = \frac{\text{param vol commensurate with } \mathcal{D}}{\text{param vol commensurate with any data}}, \tag{45}$$

is the ratio of two volumes in parameter space: the volume that can account for data $\mathcal{D}$ and the prior volume, i.e., accessible to the model without regard to $\mathcal{D}$. The Occam factor has magnitude less than 1.0; it is simply the factor by which the hypothesis space collapses by the presence of data.

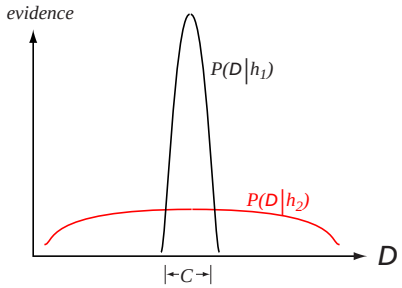


Figure 9.12: A complex model, for instance one having many free parameters, here h_2, can generate to a very wide range of data than can a more model h_1 having fewer parameters. The evidence for model h_i, is $P(\mathcal{D}|h_i)$. Suppose the two models h_1 and h_2 have been assigned the same prior probabilities. If the data in fact lies in the range C, marked, then the less powerful model h_1 is the more probable model, and should be chosen. Of course, after model h_1 has been selected in this way, its parameters $\boldsymbol{\theta}$ need to be set, for instance by standard maximum likelihood methods.

As mentioned, there may be degeneracies in our model — several parameters could be relabelled and the likelihood unchanged. This, we shall see, is especially common in neural network models where our parameterization comprises many equivalent weights (Chap **??**). For such cases, we must multiply the right hand side of Eq. 44 by the degeracy of $\hat{\boldsymbol{\theta}}$ in order to obtain the proper estimate of the evidence.

OCCAM FACTOR

If $\boldsymbol{\theta}$ is k-dimensional, and the posterior can be approximated by a Gaussian, then the *Occam factor* can be calculated directly (Problem **??**), yielding:

$$P(\mathcal{D}|h_i) \simeq \underbrace{P(\mathcal{D}|\hat{\boldsymbol{\theta}}, h_i)}_{\substack{\text{best fit} \\ \text{likelihood}}} \underbrace{P(\hat{\boldsymbol{\theta}}|h_i)(2\pi)^{k/2}|\mathbf{H}|^{-1/2}}_{\text{Occam factor}}. \tag{46}$$

where $\mathbf{H} \equiv \nabla\nabla \ln p(\boldsymbol{\theta}|\mathcal{D}, H_i)$, is a Hessian matrix — a matrix of second-order derivatives — a measure of how "peaked" the posterior is around the MAP value. Note that this Gaussian approximation does not rely on the fact that the underlying *model* $p(\mathbf{x}|\omega)$ is or is not Gaussian. Rather, it is based on the law of large numbers, independent uncorrelated processes, such as the selection of data, noise, etc.

The posterior probability of each model is given by

$$P(h_i|\mathcal{D}) \propto P(\mathcal{D}|h_i)P(h_i), \tag{47}$$

and we select the model having the highest such posterior. This procedure is itself not Bayesian; a Beayesian procedure would average overa all possible models when making a decision. The evidence for h_i, i.e., $P(\mathcal{D}|h_i)$, was ignored in a maximum likelihood setting of parameters $\hat{\boldsymbol{\theta}}$; nevertheless it is the centeral term in our comparison of

models. In practice, the evidence terms in Eq. 47 dominates the prior term, and it is traditional to ignore such priors (which are often highly subjective anyway). The algorithm below illustrates the method.

Figure 9.13 shows xxx

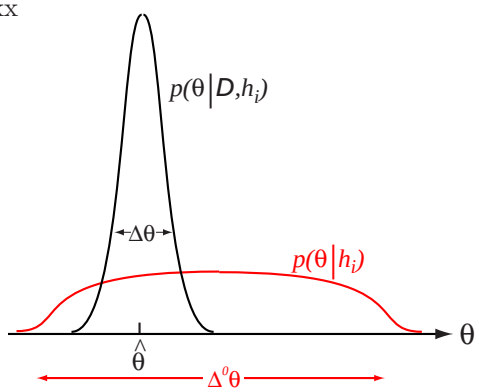


Figure 9.13: Occam Factor figure

Most of the arguments in Sect. **??** apply analogously to maximum likelihood and Bayesian model selection. In practice, though, the integration inherent in Bayesian methods is simplified using a Gaussian approximation to the evidence. Since calculating the needed Hessian via differention is nearly always simpler than a high-dimensional integration, the Bayesian method of model selection is not at a severe computational disadvantage relative to its maximum likelihood counterpart.

9.8 *Capacity and Vapnik-Chervonenkis Dimension

Growth function

$$\text{Prob}\ (\max_f |g_p(f) - g(f)|) > \epsilon) \leq 4m(2p)e^{-\epsilon^2 p/8} \tag{48}$$

9.8.1 The VC Theorem

[[??]]

9.8.2 The Capacity of a Separating Plane

The importance of having an overdetermined solution is as significant for classification as it is for estimation. For a relatively simple example, consider the partitioning of a d-dimensional feature space by a hyperplane $\mathbf{w}^t\mathbf{x} + w_0 = 0$, as might be trained by the Perceptron algorithm (Chap. **??**). Suppose that we are given n sample points in general position (i.e., with no subset of $d+1$ points falls in a $(d-1)$-dimensional subspace). Assume each point is labelled either ω_1 or ω_2. Of the 2^n possible dichotomies of n points in d dimensions, a certain fraction $f(n,d)$ are said to be linear dichotomies. These are the labellings for which there exists a hyperplane separating the points labelled ω_1 from the points labelled ω_2. It can be shown (Problem **??**) that this fraction is given by

$$f(n,d) = \begin{cases} 1 & n \leq d+1 \\ \frac{2}{2^n} \sum_{i=0}^{d} \binom{n-1}{i} & n > d+1. \end{cases} \tag{49}$$

This function is plotted in Fig. **??** for several values of d. Note that all dichotomies of $d+1$ or fewer points are linear. This means that a hyperplane is not overconstrained by the requirement of correctly classifying $d+1$ or fewer points. In fact, if d is large it is not until n is a sizeable fraction of $2(d+1)$ that the problem begins to become difficult. At $n = 2(d+1)$, which is sometimes called the *capacity* of a hyperplane, half of the possible dichotomies are still linear. Thus, a linear discriminant is not effectively overdetermined until the number of samples is several times as large as the dimensionality. This is often expressed as: "generalization begins only after learning ends."

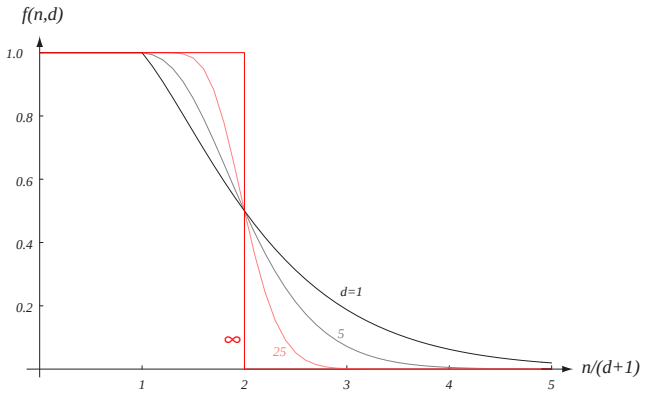


Figure 9.14: See Eq. 49.

9.8.3 The Problem-Average Error Rate

The examples we have given thus far suggest that the problem with having only a small number of samples is that the resulting classifier will not perform well on new data — it will not generalize. Thus, we expect the error rate to be a function of the number n of training samples, typically decreasing to some minimum value as n approaches infinity. To investigate this analytically, we must carry out the following steps:

1. Estimate the unknown parameters from samples.
2. Use these estimates to determine the classifier.
3. Calculate the error rate for the resulting classifier.

In general this analysis is very complicated. The answer depends on everything — on particular samples obtained, on the way they are used to determine the classifier, and on the unknown, underlying probability structure. However, by using histogram approximations to the unknown probability densities and averaging appropriately, it is possible to draw some interesting conclusions.

Consider the two-class case in which the two classes are equally likely a priori. Suppose that we partition the feature space into some number m of disjoint cells

$\mathcal{C}_1, ..., \mathcal{C}_m$. If the conditional densitites $p(\mathbf{x}|\omega_1)$ and $p(\mathbf{x}|\omega_2)$ do not vary appreciably within any cell, then instead of needing to know the act value of $\mathbf{x}$, we need only know into which cell $\mathbf{x}$ falls. This reduces the problem to the discrete case. Let $p_i = P(\mathbf{x} \in \mathcal{C}|\omega_1)$ and $q_i = P(\mathbf{x} \in \mathcal{C}|\omega_2)$. Then, since we have assumed that $P(\omega_1) = P(\omega_2) = 1/2$, the vectors $\mathbf{p} = (p_1, ..., p_m)^t$ and $\mathbf{q} = (q_1, ..., q_m)^t$ determine the probability structure of the problem. If $\mathbf{x}$ falls in $\mathcal{C}_i$, the Bayes decision rule is to decide ω_i if $p_i > q_i$. The resulting Bayes error rate is given by

$$P(e|\mathbf{p}, \mathbf{q}) = \frac{1}{2} \sum_{i=1}^{m} \min\,[p_i, q_i] \tag{50}$$

When the parameters $\mathbf{p}$ and $\mathbf{q}$ are unknown and must be estimated from a set of samples, the resulting error rate will be larger than the Bayes rate. The exact answer will depend on the set of training samples and the way in which they are used to obtain the classifier. Suppose that half of the samples are labelled ω_1 and half are labelled ω_2, with n_{ij} being the number that fall in $\mathcal{C}_i$ and are labelled ω_j. Suppose fuirther that we design the classifier by using the maximum likelihood estimates $\hat{p}_i = 2n_{i1}/n$ and $\hat{q}_i = 2n_{i2}/n$ as if they were the true values. Then a new feature vector falling in $\mathcal{C}_i$ will be assigned to ω_1 if $n_{i1} > n_{i2}$. With all of these assumptions, it follows that the probability of error for the resulting classifier is given by

$$P(e|\mathbf{p}, \mathbf{q}, \mathcal{H}) = \frac{1}{2} \sum_{n_{i1} > n_{i2}} q_i + \frac{1}{2} \sum_{n_{i1} \leq n_{i2}} p_i. \tag{51}$$

To evaluate this probability of error, we need to know the true conditional probabilities $\mathbf{p}$ and $\mathbf{q}$, and the set of training samples, or at least the numbers n_{ij}. Different sets of n random samples will yield different values for $P(e|\mathbf{p}, \mathbf{q}, \mathcal{H})$. We can use the fact that the numbers n_{ij} have a multinomial distribution to average over all of the possible sets of n random samples and obtain an average probability of error $P(e|\mathbf{p}, \mathbf{q}, n)$. Roughly speaking, this is the typical error rate one should expect for n samples. However, evaluation of this average error rate still requires knowing the underlying problem, i.e., the values for $\mathbf{p}$ and $\mathbf{q}$. If $\mathbf{p}$ and $\mathbf{q}$ are quite different, the average error rate will be near zero, while if $\mathbf{p}$ and $\mathbf{q}$ are quite similar, it will be near one-half.

A sweeping way to eliminate this dependence of the answer on the problem is to average the answer over all possible problems! That is, we assume some a priori distribution for the unknown parameters $\mathbf{p}$ and $\mathbf{q}$, and average $P(e|\mathbf{p}, \mathbf{q}, n)$ with respect to $\mathbf{p}$ and $\mathbf{q}$. The resulting *problem-average probability of error* $\bar{P}(e|m, n)$ will depend only on the number m of cells, the number n of samples, and the a priori distribution.

Of course, choosing the a priori distribution is a delicate matter. By favoring easy problems, we can make $\bar{P}$ approach zero, and by favoring hard problems we can make $\bar{P}$ approach one-half. We would like to choose an a priori distribution corresponding to the class of problems we typically encounter, but there is no obvious way to do that. A bold approach is merely to assume that problems are "uniformly distributed," i.e., that the vectors $\mathbf{p}$ and $\mathbf{q}$ are distributed uniformly over the simplexes $p_i \geq 0, \sum_{i=1}^{m} p_i = 1, q_i \geq 0, \sum_{i=1}^{m} q_i = 1$. G. F. Hughes, who suggested this approach, actually carried out the required computations and obtained the results shown graphically in Figure ??. Let us consider some of the implications of these results.

Note first that the curves show $\bar{P}$ as a function number of cells for a fixed number of training samples. With an infinite number of the maximum likelihood estimates

are perfect, and $\bar{P}$ is the average of the Bayes error rate over all problems. The corresponding curve for $\bar{P}(e|m,\infty)$ decreases rapidly from 0.5 at $m = 1$ to the asymptotic value of 0.25 as m approaches infinity. The fact that $\bar{P} = 0.5$ if $m = 1$ is not surprising, since if there is only one cell the decision must be based solely on the a priori probabilities. The fact that it is halfway between the extremes of 0.0 and 0.5. The fact that the problem-average error rate is so high merely shows that many hopelessly difficult classification problems are included in this average. Clearly, it would be rash indeed to conclude that the "average" pattern recognition problem will have this error rate.

However, the most interesting feature of these curves is that for every curve involving a finite number of samples there is an optimal number of cells. This is directly related to the fact that with a finite number of samples the performance will worsen if too many features are used. In this case it is clear why this occurs. At first, increasing the number of cells makes it easier to distinguish between $p(\mathbf{x}|\omega_1)$ and $p(\mathbf{x}|\omega_2)$ (as represented by the vectors $\mathbf{p}$ and $\mathbf{q}$), thereby allowing improved performance. However, if the number of cells becomes too large, there will not be enough training samples to fill them. Eventually, the number of samples in most cells will be zero, and we must return to using just the ineffective a priori probabilities for classification. Thus, for any finite n, $\bar{P}(e|m,n)$ must approach 0.5 as m approaches infinity.

The value of m for which $\bar{P}(e|m,n)$ is minimum is remarkably small. For $n = 500$ samples, it is somewhere around $m = 200$ cells. Suppose that we were to form the cells by dividing each feature axis into l intervals. Then with d features we would have $m = l^d$ cells. If $l = 2$, which is extremely crude quantization, this implies that using more than four or five binary features will lead to worse rather than better performance. This is a very pessimistic result, but then so is the statement that the average error rate is 0.25. These numerical values are a consequence of the a priori distribution chosen for the problems, and are of no significance when one is facing a particular problem. the main thing to be learned from this analysis is that the performance of a classifier certainly does depend on the number of design samples, and that if this number is fixed, increasing the number of features beyond a certain point is likely to be counterproductive

9.8.4 VC dimension and multi-layer neural networks

[[??]]

9.9 Combining classifiers

We have already mentioned classifiers whose decision is based on the outputs of component classifiers (Sects. **??** & **??**. Such classifiers are variously called mixture of expert models, ensemble classifiers or occasionally pooled classifiers.

The goal is to maximize the log likelihood, or (using the notation in Nowlan's overview)

$$\max \log L = \sum_c log(P(\mathbf{d}_c|\mathbf{x}_c) \tag{52}$$

where $\mathbf{x}_c$ is the input to network c. The error for single expert i is

$$E_i = p(E_i|\mathbf{d}_c)(\mathbf{d}_c - \mathbf{o}_i), \tag{53}$$

and the errors for the gating net are

$$E_g(\mathbf{x}_i)p(E_i|\mathbf{d}_c) - p_i. \tag{54}$$

Our goal is to maximize:

$$\max \log L = \sum_c logP(\mathbf{d}_c) \tag{55}$$

where

$$P(\mathbf{d}_c) = \sum_{i=1}^{N} p_i p(\mathbf{d}_c|i) \tag{56}$$

The probability of expert j generating the desired output is:

$$p(\mathbf{d}_c|j) = \frac{1}{K\sigma_j} e^{-\frac{||\mathbf{d}_c - \mathbf{o}_j(\mathbf{x}_c||^2}{2\sigma_j^2}} \tag{57}$$

We gate the individual expert networks via soft max (Sec. **??**), i.e.,

$$p_j(\mathbf{I}'_c = \frac{e^{x_j^p}}{\sum_i e^{x_i^p}}. \tag{58}$$

The derivatives for the expert networks are:

$$\frac{\partial \log L}{\partial \mathbf{O}_j} = \frac{1}{\sigma_j^2} \sum_c p(j|\mathbf{d}_c)(\mathbf{d}_c - \mathbf{o}_j) \tag{59}$$

where

$$p(j|\mathbf{d}_c) = \frac{p_j p(\mathbf{d}_c|j)}{\sum_i p_i p(\mathbf{d}_c|i)} \tag{60}$$

For the gating network we have

$$\frac{\partial \log L}{\partial x_j^p} = \sum_c p(j|\mathbf{d}_c) - p_j. \tag{61}$$

The goal is to minimize the correlation between the experts. One way: use different inputs for the gating network than for the expert networks. Alternatively, provide different inputs to the experts. Multimodal sensory integration, such as speechreading, where the experts are visual and acoustic subsystems.

The superiority of mixture of expert models seems to fly in the face of Occam motivation: keep the solution simple. In fact, the models are *not* simple!

Before winner take all, now more gradual combination

The basic notion is to have a set of "experts" — classifiers that perform well on a subset of the data. (In fact, the methodology is more general, but it is most successful for problems where component classifiers work well in limited domains. Sometimes called *mixture of experts*

MIXTURE OF EXPERTS

Why does it work?

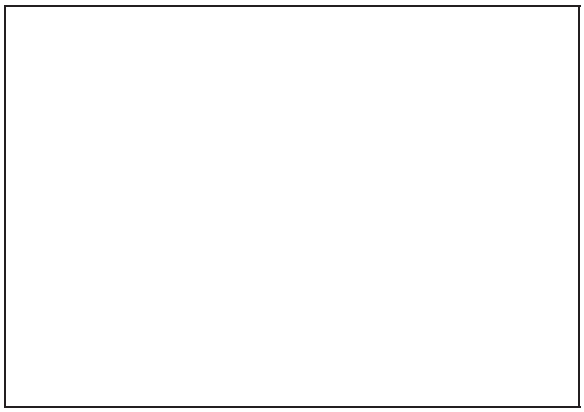


Figure 9.15: Basic combination of classifiers. The expert networks are gated by a gating network — a convex combination of the experts.

9.9.1 Bias and variance of mixture of experts

Mixture of experts classifiers yield a target output as a linear combination of the estimates of the component experts, and thus the bias and variance can be written in terms of the gating network's linear coefficients and the expert outputs of the experts.

The variance is thus:

$$\begin{aligned} E_D[(\mu - E_D[\mu])^2] &= \underbrace{E_D\left[\sum_j (g_j\mu_j - E_D[g_j\mu_j])^2\right]}_{var\ of\ experts} \\ &+ \underbrace{E_D\left[\sum_j \sum_{k\neq j} (g_i\mu_i - E_D[g_j\mu_j])(g_k\mu_k - E_D[g_k\mu_k])\right]}_{cov\ of\ experts'\ outputs}. \end{aligned} \tag{62}$$

9.9.2 Gaussian mixtures

Assume a normal theory of errors, i.e., a model $f(\mathbf{x}|\boldsymbol{\theta}_i) \sim N(\mathbf{x}, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$. Assume mean is zero (obtained by a trivial transformation). All experts are calibrated or unbiased.

Assume the decision maker's prior distribution for $\boldsymbol{\theta}$ is normal with mean m_0 and variance σ_0^2. The estimation of error $m_0 - \theta$ is uncorrelated with any of the experts's errors.

The decision maker consults k experts, the prosterior distribution of θ is a normal distribution with mean m_* and variance σ_*^2 (Problem 33):

$$\begin{aligned} m_* &= \left[\frac{m_0}{\sigma_o^2} + \mathbf{e}^t\boldsymbol{\Sigma}^{-1}\mathbf{x}\right]\sigma_*^2, \\ \sigma_*^2 &= \left[\frac{1}{\sigma_0^{-2}} + \mathbf{e}^t\boldsymbol{\Sigma}^{-1}\mathbf{e}\right]^{-1}, \end{aligned} \tag{63}$$

where $\mathbf{e} = (1, 1, \ldots, 1)$.

Surely we have a better estimate than with a single expert (except in pathological cases). We now turn to quantifying this improvement.

Suppose the decision maker could consult n independent experts, each of which has a variance of σ^2. We can describe the prior variance σ_0^2 of the decision maker in the form σ^2/r_0, where r_0 can be considered an equivalent sample size for determining prior distribution. Then, after consulting r independent experts, the posterior variance would be

$$\sigma_*^2 = \frac{\sigma^2}{n_0 + n} = \frac{1}{\sigma_0^{-2} + n\sigma^{-2}}. \tag{64}$$

Now, let us acknowledge that typically the experts are not independent. Equation 63 githe posterior variance for a multinormal ditribution having covariance $\mathbf{\Sigma}$. The value of n xperts that have the same variance σ^2 as with k experts having covariance $\mathbf{\Sigma}$ is the "equivalent number of independent experts" wth variance σ^2, and is denoted $m(\sigma^2, \mathbf{\Sigma})$.

From Eqs. 63 & 64 we find

$$n(\sigma^2, \mathbf{\Sigma}) = \sigma_2 \mathbf{e}\mathbf{\Sigma}^{-1}\mathbf{e}. \tag{65}$$

Common correlation, common variance

Suppose $\mathbf{\Sigma}$ is an intraclass correlation matrix; thus all variances are equal and all correlations are equal, σ^2 and $\rho > 0$, respectively. This implies that the experts are exchangeable. Then

$$\mathbf{e}^t\mathbf{\Sigma}^{-1}\mathbf{e} = \frac{k}{\sigma^2}\frac{1}{1 + (k-1)\rho}, \tag{66}$$

and thus

$$n(\sigma_2, \mathbf{\Sigma}) = \frac{k}{1 + (k-1)\rho}. \tag{67}$$

Thus, the n independent and the k dependent experts yd the same error variance σ^2. For $k > 1$, we have $n(\sigma^2, \mathbf{\Sigma}) < k$, that is, positive dependence will reduce the information content of a set of experts' estimates and the stronger the dependence, the greater the reduction (Problem 34).

Common correlation, different variances

Suppose not that there is a common correlation, $\rho > 0$, but the error variances differ. Then we have

$$\mathbf{e}^t\mathbf{\Sigma}^{-1}\mathbf{e} = \frac{[1 + (k-1)\rho]\sum_{j=1}^{k}\sigma_j^{-2} - \rho(\sum_{j=1}^{k}\sigma_j^{-1})^2}{(1-\rho)[1 + (k-1)\rho]}. \tag{68}$$

If we assume that $\rho < [1 - k + \sigma_i \sum_{j=1}^{k}\sigma_j^{-1}]^{-1}$ insures only positive weights (Problem 38).

Arbitrary covariance

The general case has difference covariances σ_i^2 and different correlations ρ_{ij} between errors. We assume $\alpha_i = (\mathbf{e}^t \mathbf{\Sigma}^{-1})_i > 0$ for all i, which insures positive weights.

We have

$$\frac{d(\mathbf{e}\mathbf{\Sigma}^{-1}\mathbf{e})}{d\rho_{ij}} = \left(\frac{d(\mathbf{e}\mathbf{\Sigma}^{-1}\mathbf{e})}{d\sigma_{ij}}\right)\left(\frac{d\sigma_{ij}}{d\rho_{ij}}\right) = \sigma_i\sigma_j\left(\frac{d(\mathbf{e}\mathbf{\Sigma}^{-1}\mathbf{e})}{d\sigma_{ij}}\right). \tag{69}$$

$$n \geq \frac{1}{\epsilon}\left(\ln\frac{1}{\delta} + \ln|\mathcal{H}|\right). \tag{70}$$

n is the "sample complexity"

9.9.3 Gaussian Priors

$O(\frac{n^4}{\epsilon^3} \cdot \ln \frac{n}{\delta})$

9.9.4 Relation to Bayesian Learning

[[??]]

Uniform convergence

$$P(\underset{\mathbf{w} \in \mathbf{W}}{sup}|E_G(\mathbf{w}) - E_L(\mathbf{w})| > \epsilon) \rightarrow 0 \text{as} N \rightarrow \infty. \tag{71}$$

where sup is the supremum.

$$E_G(\mathbf{w}) \leq E_L(\mathbf{w}) + C\left(\frac{N}{VC_{dim}}, E_L(\mathbf{w}), \eta\right), \tag{72}$$

where $C(\cdot)$ is the confidence interval, and obeys

$$C\left(\frac{N}{VC_{dim}}, E_L(\mathbf{w}), \eta\right) = 2\Psi\left(\frac{N}{VC_{dim}}, \eta\right)\left(1 + \sqrt{1 + \frac{E_L}{\Psi(\frac{N}{VC_{dim}}, \eta)}}\right), \tag{73}$$

where

$$\Psi\left(\frac{N}{VC_{dim}}, \eta\right) = \frac{1}{N}\left((ln(\frac{2N}{VC_{dim}} + 1)VC_{dim} - ln\eta)\right) = \epsilon^2. \tag{74}$$

and

The assumptions: large training set sizes (though in practice can work with intermediate ones) The theory holds for linear decision regions, but in practice also for complex ones, such as multi-layer neural nets. Large number of weights. Need not have zero error in asymptote. Technical assumptions lead to $1/2 \leq \alpha \leq 1$.

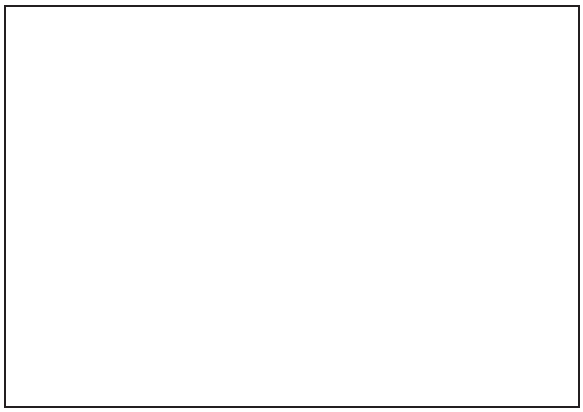


Figure 9.16: The generalization error, the learning error and the confidence interval as a function of the number of training patterns.

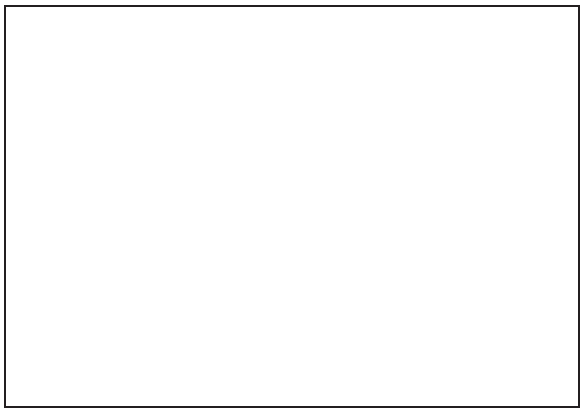


Figure 9.17: Generalization and learning errors and confidence interval as a function of the capacity, which is related to the number of degrees of freedom.

Summary

The No Free Lunch theorem states that there are no a priori reasons for preferring one learning algorithm to another in the absense of information about the problem. There are as many priors in which learning algorithm 1 outperforms algorithm 2 as vice versa (even if algorithm 2 is random guessing or a constant output). More specifically, in the absence of prior information cross-validation is as successful as anti-cross-validation, pruning as non-pruning, stopped training as non-stopped training. The theorems force us to acknowledge the importance of priors and added information. Given that we use a finite set of feature values that enable distinguishing any to patterns under consideration the Ugly Duckling theorem states that the number of predicates shared by any two different patterns under consideration is constant, and does not depend upon the choice of the two objects. Together, these theorems highlight the need for insight into proper features and matching the algorithm to the data distribution — there is no problem independent "best" learning or pattern recognition system nor feature representation. A corollary of the No Free Lunch Theorem is that formal theory and algorithms taken alone are not enough for pattern classification; learning is an empirical subject.

The basic insight underlying resampling techniques — such as the bootstrap, jackknife, cross-validation, boosting, bagging methods — is that multiple

The bias-variance relation states how prior knowledge or constraints upon the form of the solution (bias) affects the variability in the output (variance). Ideally we seek a class of solutions that contain the true function, with as few free parameters as possible.

The minimum description length principle xxx. More sophisticated versions can be found in original work by Chaitin and Rissanen, among others. Schwarz described a $k/2\log b$ penalty on the maximum likelihoodm where

AIC, Akaike Information criterion, ...

Learning curve

There are a number of methods for combining the outputs of separate "expert" classifiers, such as linear weighting, winner-takes-all, and so on.

Bibliographical and Historical Remarks

Wolpert's book [80] has a number of papers on the foundations of the theory of generalization, and presentations of the No Free lunch theorems appear in [82] and the analysis of the Bayesian Occam's factor for Occam's razor [81]. (The clever name for these theorems is due to David Haussler.) Zhu and Rohwer demonstrated the no free lunch theorems for cross-validation [84]. The Ugly Duckling theorem appears in [77], which also explores some of its philosophical implications [57].

Variations on Kolmogorov Complexity

Occam's razor

Akaike Information Criterion (AIC) [1, 2]

Gideon Schwarz [63]

It is worth pointing out alternatives to Occam's razor. For instance, principle of multiple explanations: If several theories are consistent with the data, retain all such theories.

Bayes: the probabilitiey of a model or hypothesis being true is proportional to the designer's prior belief in the hypothesis multipleied by the conditional probability of

the data given the hypothesis in question.

Newton: in **Principia** [52] "Natura enim simplex est, et rerum causis superfluis non luxuriat," or "for nature indeed is simple, and does not luxuriate in superfluous causes."

Epicurus (342?–270?BC), in a letter to Pythocles, stated the principle of multiple explanations, claiming that [22], which we now might call the principle of indifference

Bayes said to include prior knowledge.

Stacked generalization [67, 68]

Karl Popper has argued that Occam's razor is without operational value, since there is not clear criterion or measure of simplicity [54]

Efron's clear book gives the theoretical foundations of some of the most important resampling techniques [21], and papers on bootstrap techniques for error estimation include [35]. Quenouille introduced the term jackknife [56]. Breiman has been particularly active an suggesting resampling methods, such as bagging [7] and arcing [8].

Cross validation was introduced by Cover [17].

The zero-one loss is central for classification, but if the loss function is quadratic, then averaging over all targets does not give equal performance to all algorithms. In fact, there is a benefit in guessing near the middle of the range of possible outputs [80].

Schapire strength of weak learnability [61] and Yoav Freund early work [23]

Bias and variance

Important work on learning curves: [16]

Geman et al[28]

On bias, variance, 0/1-loss and the curse-of-dimensionality [25]

The roots of maximum likelihood model selection stem from Bayes himself, but were originally explored by [29]. Interest in Bayesian model selection was revived in a series of papers by MacKay, whose primary interest was in applying the method to neural networks and interpolation [48, 51, 50, 49]. These model selection methods have subtle relationships to minimum description length (MDL) [58] and maximum entropy approaches — topics that would take us a bit beyond the depth here. An empirical study showing that simple classifiers often work well can be found in [33].

FukunagaHayes[27]

A fairly complete book on techniques for combining neural nets is [65] and classifiers more generally [39, 40]. Perrone and Cooper analyzed the case when expert networks disagree [53].

Adaboost [24], which Breiman calls this arc-sf.

Boosting in multicategory problems is a bit more subtle than two-category problems [62]

RaudysJain[76]

Deev [18]

Valiant [75]

Wymanetal[83]

Extending bais/variance to non-quadratic (as in Friedman) [31].

Schaffer too proved a "conservation law in generalization" i.e., for every possible learning algorithm for binary classification the sum of performance over all possible target functions is exactly zero (just like for random classification) [60], which was the inspiration for Fig. 9.1.

While pattern recognition practioners tend to focus on biasing the solution in feature regions of interest, because of conservation laws, it reduces performance elsewhere.

"Simplicity" in classifiers can be motivated by considering the cost (difficulty) of designing the classifier. Bounded rationality is the principle that we often settle for an adequate solution, not necessarily the optimal [66]

Use of learning with queries Open Mind [71, 73, 72]

Empirical evaluation techniques in computer vision [6]

The foundational work on Kolmogorov complexity appears in [41, 42, 69, 70], but a short elementary overview [9] and particularly Chaitin's [10] Li and Vitányi's[47] books are far more accessible. Barron and Cover's [4]

First Corinna Cortes paper on estimating the final quiality of a classifier [15] (heuristic)

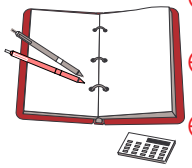
Cohn and his colleagues have been active in active learning [12, 14]

CohnTesauro[13]

Anoulovaetal[3]

SchwarzeHertz[64]

Problems



⊕ *Section 9.1*

⊕ *Section 9.2*

⊕ *Section 9.3*

⊕ *Section 9.5*

⊕ *Section 9.6*

⊕ *Section 9.7*

⊕ *Section 9.8*

⊕ *Section 9.9*

1. Prove part *1* of Theorem 9.1, i.e., that uniformly averaged over all target functions F, $\mathcal{E}_1(E|F,n) - \mathcal{E}_2(E|F,n) = 0$. xxxx

2. Prove part *2* of Theorem 9.1, i.e., for any fixed training set $\mathcal{D}$, uniformly averaged over F, $\mathcal{E}_1(E|F,\mathcal{D}) - \mathcal{E}_2(E|F,\mathcal{D}) = 0$. xxxx

3. Prove part *3* of Theorem 9.1, i.e., uniformly averaged over all priors $P(F)$, $\mathcal{E}_1(E|n) - \mathcal{E}_2(E|n) = 0$. Summarize and interpret the part in words. xxxx

4. Prove part *4* of Theorem 9.1, i.e., for any fixed training set $\mathcal{D}$, uniformly averaged over $P(F)$, $\mathcal{E}_1(E|\mathcal{D}) - \mathcal{E}_2(E|\mathcal{D}) = 0$. Summarize and interpret the part in words. xxx

5. Consider that a hypothesis space N has finite VC dimension. Sauer's Lemma is a statement about the growth function $\mathcal{G}(m)$:

$$\mathcal{G}(m) \leq 1 + \binom{m}{1} + \binom{m}{2} + \cdots + \binom{m}{d} \equiv \Phi(d,m),$$

where $\binom{m}{k}$ is the binomial coefficient.

Do this as follows:

(a) Prove the identity $\Phi(d,m) = \Phi(d,m-1) + \Phi(d-1,m-1)$ for $d \geq 1$ and $m \geq 2$.

(b) Assume that the VC dimension is 0. Confirm that Sauer's Lemma is true.

(c) Use induction on $d+m$. (If necessary, make natural simplifying assumptions, for instance that there are no repeated training points, or inconsistently labelled points.)

6.

Expand the left hand side of Eq. 9 to get the right hand size, which expresses the error as a sum of a $bias^2$ plus *variance*.

7. Prove that the average of the leave one out means, $\mu_{(\cdot)}$ (Eq. **??**), is equal to the sample mean μ (Eq. 20).

8. The No Free Lunch Theorem implies that cross validation must fail as often as it succeeds. Show this as follows: Consider algorithm A to be *anti-cross validation*, which advocates choosing the model that does *worst* on a validation set, and algorithm B to be standard cross validation, both used in conjunction with some standard classification method.

(a) Argue that if cross validation were better than anti-cross validation overall, the NFL theorem would be violated.

(b) xxx

9. Use the NFL Theorem to explain why there can be no model independent way to estimate the final classification accuracy before a classifier has been trained. Consider how some purported model independent method allowed us to select between two algorithms A and B, and perform better overall.

10. Suppose you call an algorithm better if it performs slightly better than average over *most* problems, but very poorly on a small number of problems. Explain why the NFL Theorem does not preclude the existence of algorithms "better" in this way.

11.

As in Problem , but more general.

12. Using the definitions and assumptions in Sect. **??**, prove that Y_F and Y_H are conditionally independent given d and q.

(a) Expand $P(y_F, y_H|d,q)$ and show it is equal to $P(y_f)|d,q)P(y_H|d,q)$.

(b) Interpret this significance of this.

⊕ *Section 9.7*

13. Suppose we believe that the data for a pattern classification task from one category comes either from a uniform distribution $p(x) \sim U(x_l, x_u)$ or from a normal distribution, $p(x) \sim N(\mu, \sigma^2)$, but we have no reason to prefer one over the other. Our sample data is $\mathcal{D} = \{.2, .5, .4, .3, .9, .7, .6\}$.

(a) Find the maximum likelihood values of x_l, and x_u for the uniform model.

(b) Find the maximum likelihood values of μ and σ for the Gaussian model.

(c) Use maximum likelihood model selection to decide which model should be preferred.

14. Suppose we believe that the data for a pattern classification task from one category comes either from a uniform distribution bounded below by 0, i.e., $p(x) \sim U(0, x_u)$ or from a normal distribution, $p(x) \sim N(\mu, \sigma^2)$, but we have no reason to prefer one over the other. Our sample data is $\mathcal{D} = \{.2, .5, .4, .3, .9, .7, .6\}$.

(a) Find the maximum likelihood values of x_u for the uniform model.

(b) Find the maximum likelihood values of μ and σ for the Gaussian model.

(c) Use maximum likelihood model selection to decide which model should be preferred.

(d) State qualitatively the difference between your solution here and that to Problem 13, without necessarily having to solve that problem. In particular, what are the implications from the fact that the two candidate models have different numbers of parameters?

15. Consider three candidate one-dimensional distributions each parameterized by an unknown value for its "center":

- Gaussian: $p(x) \sim N(\mu, 1)$
- Triangle: $p(x) \sim T(\mu,1) = \begin{cases} 1 - |x - \mu| & \text{for } |x - \mu| < 1 \\ 0 & \text{otherwise} \end{cases}$
- Uniform: $p(x) \sim U(\mu - 1, \mu + 1)$.

We are given the data $\mathcal{D} = \{-0.9, -0.1, 0., 0.1, 0.9\}$, and thus, clearly the maximum likelihood solution $\hat{\mu} = 0$ applies to each model.

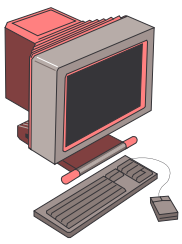
(a) Use maximum likelihood model selection to determine the best model for this data. State clearly any assumptions you make.

(b) Suppose we are sure for each model that the center must lie in the range $-1 < \mu < 1$. Calculate the Occam factor for each model and the data given.

(c) Use Bayesian methods to select the best model for $\mathcal{D}$.

16. Recall that an estimator is *consistent* if it convergest to the true value in the limit of infinite training data. Prove that the mean is *not* for the distribution $p(x) \sim \tan^{-1}(x - a)$ for a some finite real constant.

Computer exercises



1.
Write a program to lkjsdflksdj lkjsdflsdj sdlkfjsd fsd dlfkjs df

$$p(x|\omega_i) = \frac{1}{\pi b} \cdot \frac{1}{1 + \left(\frac{x - a_i}{b}\right)^2}, \qquad i = 1, 2.$$

$$\begin{aligned} p_{i1} = &\quad p > 1/2 \qquad i = 1, ..., d \\ p_{i2} = &\quad 1 - p \qquad i = 1, ..., d \end{aligned}$$

17.
Verify that Eq. **??** gives the same answer as Eq. 21.

18.
Consider the computational complexity of the jacknife and bootstrap estimates of a one-dimensional data set of n points.

(a) The jackknife estimate of the mean.

(b) The jackknife estimate of the median.

(c) The jackknife estimate of the standard deviation.

(d) The bootstrap estimate of the mean.

(e) The bootstrap estimate of the median.

(f) The bootstrap estimate of the standard deviation.

19.)
Assume a cost function $c = \sum_Y [f(q, y) - h(q, y)]^2$, derive Eq. **??**.

20. One of the "conservations laws" for generalization states that the positive generalization performance of an algorithm in some learning situations must be offset by negative performance elsewhere. Consider a very simple learning algorithm that seems to contradict this law. For each test pattern, the prediction of the *majority learning algorithm* is merely the category most prevalent in the training data. When two categories are equally likely, this algorithm

(a) lksjdf

(b) lksjdf

21. Repeat Problem 20 but for the *minority learning algorithm*, which always predicts the category label of the category *least* prevalent in the training data.

22.
Starting from Eq. **??**

(a) If f is single-valued at q, then $\sigma^2_{f,m,q} = 0$

(b) $\sigma^2_{f,m,q}$ is independent of $P(h|d)$

(c) $\sigma^2_{f,m,q}$ is a lower bound on the error — it is the Bayes error

(d) If average guess of the algorithm is the average guess of the target, then $bias_{f,m,q} = 0$

(e) $variance_{f,m,q} \geq 0$ and is $= 0$ if the algorithm's guess is independent of d.

(f) $variance_{f,m,q}$ depends on f only indirectly, through $P(d|f, m)$

(g) All terms are continuous in f. This is desirable.

23.).
Consider classifiers.

(a) Show by simple example of a two-category problem that an order-corret classifier.

(b) Suppose we have xxx. Which of the following classifiers are order-correct at xxx? at xxx?

- xxx
- xxx
- xxx

(c) Consider a c-category classification problem. Is the Bayes classifier necessarily order-correct? If so, prove it. If not, give a simple counterexample.

24. What is the computational complexity of full jackknife estimate of accuracy and variance for an unpruned nearest-neighbor classifier?

25.).
Calculate the jackknife unbiased estimate of variance for the following problem. Our input space is the real line, and the variance $\hat{theta} = \sum_{i=1}^{n}(x_i - \bar{x})^2/(n-1)$. Let the kth central moments be

$$\mu_k = E_F[X - E_f[x]]^k \text{ and } \hat{\mu}_k = \frac{1}{k}\sum_{i=1}^{n}[x_i - \bar{x}]^k. \tag{75}$$

(a) Use a change in variable $\beta_i = (x_i - \bar{x})^2$ and Eq. **??** to show

$$var_{jk} = \frac{n^2}{(n-1)(n-2)^2}(\hat{\mu}_4 - \hat{\mu}_2^2). \tag{76}$$

(b) Show that the true variance is

$$var = \frac{\mu_4 - \frac{n-3}{n-1}\mu_2^2}{n}. \tag{77}$$

(c) compare (graph)

(d) A true functional statistic should not change if we have multiple copies of the same data. Consider $\hat{\theta}$ as calculated with a data set $\{x_1, x_2, \ldots, x_n\}$ and $\{x_1, x_1, x_2, x_2, \ldots, x_n, x_n\}$ and show that $\hat{\theta}$ is not a true functional statistic.

26. Show that with probability 1, that the probability of a linear threshold unit based on the pocket weight vector being optimal approaches 1 in the limit of infinite data.

27. Make some simple assumptions and state, using $\mathcal{O}(\cdot)$ notation, the Kolmogorov complexity of the following binary strings:

(a) $\underbrace{010110111011110\ldots}_{n}$

(b) $\underbrace{000\ldots00100\ldots000}_{n}$

(c) $e = 10.10110111111000010\ldots_2$

(d) $2e = 101.01101111110000101\ldots_2$

(e) The digits of π, but where every 100th digit is changed to the numeral 1.

(f) The digits of π, but where every nth digit is changed to the numeral 1.

28. put in a gaussian and derive Eq. 18.
29. If we use an algorithm with uniform $P(h|\mathcal{D})$, then $K(h,\mathcal{D}) = K(\mathcal{D})$ in Eq. 6. Explain and interpret.
30.

Consider a mixture of experts model with varying amounts of correlation between the component experts. Assume [make clear the assumptions here].

(a) Find the general formula given in [Clemen and Winkler, Operations Research 33 (1985) pp. 427-442]

(b) Use your result to show that two independent experts carry as much information as an infinite number of experts which have the (mutual) correlation of 0.5.

31.) Derive the jackknife estimate of the bias in Eq. 27 as follows. xxx
32. Case where MDL is easier than imposing probabilities.
33. Fill in the steps to Eq. 63.
34. Consider the case of k experts common correlation, common covariance and Eq. 67.

(a) Show that $\frac{dn(\sigma^2,\mathbf{\Sigma})}{d\rho} < 0$, and thus the stonger the statistical dependency, the greater the reduction in the value of new experts.

(b) Find the limit of an infinite number of correlated experts.

(c) Evaluate your answer for $\rho = 0.80$ and for $\rho = 0.20$. Interpret.

(d) Calculate the marginal equivalent value of the kth expert, and that it is a decreasing convex function of k.

35. In standard boosting, we seek a set: why do we want half classified correctly, rather than NONE classified correctly?
36. Fill in the steps leading to Eq. 16.
37. Prove that in the limit of $B \to \infty$, the bootstrap estimate of the variance of the mean is the same as the variance of the mean.
38. Consider the case of k experts common correlation, different covariances and Eq. 68.

(a) Show that Eq. **??** insures all experts are given positive weights.

(b) Given the inequality in Eq. **??** is obeyed, prove that

$$\frac{d(\mathbf{e}\mathbf{\Sigma}^{-1}\mathbf{e})}{d\sigma_i^{-1}} = \frac{2\left[[1+(k-1)\rho]\sigma_i^{-1} - \rho\sum_{j=1}^{k}\sigma_j^{-1}\right]}{(1-\rho)[1+(k-1)\rho]} > 0.$$

(c) Use your result and Eq. **??** to sat the smaller σ_i leads to a larger $\mathbf{e}\mathbf{\Sigma}^{-1}\mathbf{e}$, and hence a smaller posterior variance σ_*^2.

39. Derive the bootstrap estimate of the bias of Eq. 32.

40. Prove Eq. **??** as follows. Note first that

$$P(c|f,\mathcal{D}) = \sum_{y_H,y_F,q} P(c|y_H,y_H,q,f,\mathcal{D})P(y_H|y_F,q,f,\mathcal{D})P(y_F,q|f,\mathcal{D}).$$

(a) Rewrite the summand to derive

$$P(c|f,\mathcal{D}) = \sum_{y_H,y_F,q} \delta[c,L(y_H,y_F)]P(y_H|y_F,f,q,\mathcal{D})P(y_F,q|f,\mathcal{D}),$$

where, as throughout, $L(\cdot,\cdot)$ is the loss function.

(b) Integrate over hypothesis functions h to show that

$$P(y_H|y_F,f,q,\mathcal{D}) = \int P(y_H|y_f,h,f,q,\mathcal{D})P(h|y_F,f,q,\mathcal{D})\ dh = P(y_H|q,\mathcal{D}).$$

(c) Collect your results to prove the lemma.

41. Consider AdaBoost

(a) Derive Eq. 35.

(b) Let $G_k = 0.05$ for all $k = 1$ to k_{max}. Plot the upper bound on the ensemble test error given by Eq. 35.

(c) In practice, G_k often decreases. Repeat part (

42. Show by simple counterexamples that the averaging in the No Free Lunch Theorem, Theorem 9.1 must be "uniformly." For instance imagine that the sampling distribution is a Dirac delta distribution centered on a single target function, and algorithm 1 guesses the function exactly while algorithm 2 disagrees with algorithm 1 on every prediction.

43. Prove that the probability of getting a particular cost given a target function f and training set $\mathcal{D}$ is $\Lambda(c)/r$, i.e., it depends solely on the cost function itself and the number, r, of categories. Do this as follows:

(a) Explain why $p(y_H|q,\mathcal{D}) = 1/r$.

(b) Explain why, if the loss function is symmetric, on can replace $\sum_{y_H} \delta[c,L(y_H,y_F)]P(y_H|q,\mathcal{D})$ with $\lambda(c)/r$.

(c) xxx

44. from part 1, let the good algorithm be 1 and the bad predict the opposite. Then the performance of a single algorithm 1 is no better than chance, averaged over all target functions.

45.) For each of the following state whether the likelihood function is vertical or not.

(a) $P(\mathcal{D}|f) = \prod_{i=1}^{m} p[d_{\mathbf{X}}(i)]f[d_{\mathbf{X}}(i), d_{\mathbf{Y}}(i)]$.

(b) xxx

46. Prove the third No Free Lunch Theorem.

47.

State how the No Free Lunch theorems imply that you cannot use training data to distinguish between new problems for which you generalize well from those for which you generalize poorly. Argue by reductio ad absurdum: that if you could xxx then it would be conceivable that schemes like cross-validation would have that property, in which case they would violate the NFL theorems.

48.)

Prove that if recognizers' generalization does not degrade when more training data is used, then Occam's razor can be justified. (From Wolpert's *Complex Systems* paper.)

49.)

Show that $\hat{bias} = (n-1)(\hat{\theta}_{(\cdot)} - \hat{\theta})$ is unbiased statistic for estimating the true bias $E_F[\theta(\hat{F}) - \theta(F)$ as follows. First note that from Eq. **??** we have

$$\hat{\theta}_{(\cdot)} = \mu^{(n-1)} + \frac{1}{n}\sum_{1\le i\le n}\alpha_i^{(n-1)} + \frac{1}{n^2}\frac{n(n-2)}{(n-1)^2}\sum_{i\le i<i'\le n}\beta_{ii'}.$$

(a) Define $\Delta_i = \Delta(X_i) = \frac{\beta(X_i, X_i)}{2}$ and $E_F\Delta = E_F\Delta(X)$. Use these definitions to find equations for $\mu^{(n-1)} - \mu^{(n)}$ and for $\alpha^{(n-1)}(x) - \alpha^{(n)}(x)$ in terms of $E_F\Delta$ and $\Delta(x)$.

(b) Express $\hat{bias} = (n-1)(\hat{\theta}_{(\cdot)} - \hat{\theta})$ in terms of the quantities described.

(c) Use the fact that the expection of both $\Delta_i - E_F\Delta$ and $\beta_{ii'}$ vanish, prove that $E_F\hat{bias} = (E_F\Delta)/n$.

(d) Use this to complete the proof.

(see page 25 of Efron's book).

50.

Prove the Ugly Duckling Theorem, Theorem 9.2

Computer exercises

1.

⊕ *Section 9.1*

⊕ *Section 9.2*

⊕ *Section 9.3*

⊕ *Section 9.5*

⊕ *Section 9.6*

⊕ *Section 9.7*

⊕ *Section 9.8*

⊕ *Section 9.9*

Write a program to show that cross validation fails as follows. One-dimensional variable with mean zero and unit variance.

3. Consider three candidate one-dimensional distributions each parameterized by an unknown value for its "center":

- Gaussian: $p(x) \sim N(\mu, 1)$
- Triangle: $p(x) \sim T(\mu,1) = \begin{cases} 1 - |x - \mu| & \text{for } |x - \mu| < 1 \\ 0 & \text{otherwise} \end{cases}$
- Uniform: $p(x) \sim U(\mu - 1, \mu + 1)$.

We are given the data $\mathcal{D} = \{-.9, -.1, 0., .1, .9\}$, and suppose we are sure for each model that the center must lie in the range $-1 < \mu < 1$. Clearly, the maximum likelihood solution $\hat{\mu} = 0$ applies to each model.

(a) Estimate the Occam factor in each case.

(b) Use Bayesian model selection to choose the best of these models.

4. Explore the putative value of overfitting avoidance in the following simple set of problems. All cases are two categories, with six-component binary feature vectors.

(a) lskjdf

5. use MDL for training a tree

51. The "trimmed mean" of a distribution is merely the arithmetic mean of a distribution in which some percentage α (e.g., 5%) of the highest and of the lowest points have been deleted. The trimmed means is, of course, less sensitive to the presence of outliers than is the traditional mean.

(a) Prove that the trimmed mean of a distribution where $\alpha \rightarrow 0.5$ is the median.

(b) Use the (skewed) data in the table above. Bootstrap estimate.

(c) Jackknife estimate.

(d) Bootstrap bias estimate.

(e) Jackknife bias estimate.

(f) Compare to standard mean.

6. Example where cross validation need not work.

7. Basic boosting: heuristics to improve partitioning: a) just add unchosen patterns equally to the component classifiers C_k, b) adjust n_1, c) "overuse" patterns, i.e., exhaust $\mathcal{D}$, and then go back and use others again.

Bibliography

[1] Hirotugu Akaike. On entropy maximization principle. In Paruchuri R. Krishnaiah, editor, *Applications of Statistics*, pages 27–42. North-Holland, Amsterdam, The Netherlands, 1977.

[2] Hirotugu Akaike. A Bayesian analysis of the minimum AIC procedure. *Annals of the Institute of Statistical Mathematics*, 30A(1):9–14, 1978.

[3] Svetlana Anoulova, Paul Fischer, Stefan Pölt, and Hans-Ulrich Simon. Probably almost Bayes decisions. *Information and Computation*, 129(1):63–71, 1996.

[4] Andrew R. Barron and Thomas M. Cover. Minimum complexity density estimation. *IEEE Transactions on Information Theory*, IT-37(4):1034–1054, 1991.

[5] Charles H. Bennett, Péter Gács, Ming Li, Paul M. B. Vitányi, and Wojciech H. Zurek. Information distance. *IEEE Transactions on Information Theory*, IT-44(4):1407–1423, 1998.

[6] Kevin W. Bowyer and P. Jonathon Phillips, editors. *Empirical evaluation techniques in computer vision*. IEEE Computer Society, Los Alamitos, CA, 1998.

[7] Leo Breiman. Bagging predictors. *Machine Learning*, 26(2):123–140, 1996.

[8] Leo Breiman. Arcing classifiers. *The Annals of Statistics*, 26(3):801–824, 1998.

[9] Gregory J. Chaitin. Information-theoretic computational complexity. *IEEE Transactions on Information Theory*, IT-20(1):10–15, 1974.

[10] Gregory J. Chaitin. *Algorithmic Information Theory*. Cambridge University Press, Cambridge, UK, 1987.

[11] Bertrand S. Clarke and Andrew R. Barron. Information theoretic asymptotics of Bayes methods. *IEEE Transactions on Information Theory*, IT-36(3):453–471, 1990.

[12] David Cohn, Les Atlas, and Richard Ladner. Improving generalization with active learning. *Machine Learning*, 15(2):201–221, 1994.

[13] David Cohn and Gerald Tesauro. How tight are the Vapnik-Chervonenkis bounds? *Neural Computation*, 4(2):249–269, 1992.

[14] David A. Cohn, Zoubin Ghahramani, and Michael I. Jordan. Active learning with statistical models. In Gerald Tesauro, David S. Touretzky, and Todd K. Leen, editors, *Advances in Neural Information Processing Systems*, volume 7, pages 705–712, Cambridge, MA, 1995. MIT Press.

[15] Corinna Cortes, Larry D. Jackel, and Wan-Ping Chiang. Limits on learning machine accuracy imposed by data quality. In Gerald Tesauro, David S. Touretzky, and Tood K. Leen, editors, *Advances in Neural Information Processing Systems*, volume 5, pages 239–246, Cambridge, MA, 1995. MIT Press.

[16] Corinna Cortes, Larry D. Jackel, Sara A. Solla, Vladimir Vapnik, and John S. Denker. Learning curves: Asymptotic values and rate of convergence. In Jack D. Cowan, Gerald Tesauro, and Joshua Alspector, editors, *Advances in Neural Information Processing Systems*, volume 6, pages 327–334, San Francisco, CA, 1994. Morgan Kaufmann.

[17] Thomas M. Cover. Learning in pattern recognition. In Satoshi Watanabe, editor, *Methodologies of Pattern Recognition*, pages 111–132, New York, NY, 1969. Academic Press.

[18] Alexander Dimitrivich Deev. Asymptotic expansions of statistic distributions of discriminant analysis. *Statisticheskie Metody Klassifikatsii*, 1:6–51, 1972.

[19] Tom G. Dietterich. Overfitting and undercomputing in machine learning. *Computing Surveys*, 27(3):326–327, 1995.

[20] Robert P. W. Duin. A note on comparing classifiers. *Pattern Recognition Letters*, 17(5):529–536, 1996.

[21] Bradley Efron. *The Jackknife, the Bootstrap and Other Resampling Plans*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1982.

[22] Epicurus and Eugene Michael O'Connor (Editor). *The Essential Epicurus: Letters, Principal Doctrines, Vatican Sayings, and Fragments*. Prometheus Books, New York, NY, 1993.

[23] Yoav Freund. Boosting a weak learning algorithm by majority. *Information and Computation*, 121(2):256–285, 1995.

[24] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1995.

[25] Jerome H. Friedman. On bias, variance, 0/1-loss, and the curse-of-dimensionality. *Data Mining and Knowledge Discovery*, 1(1):55–77, 1997.

[26] Kenji Fukumizu. Active learning in multilayer perceptrons. In David S. Touretzky, Michael C. Mozer, and Michael E. Hasselmo, editors, *Advances in Neural Information Processing Systems*, volume 8, pages 295–301, Cambridge, MA, 1996. MIT Press.

[27] Keinosuke Fukunaga and Raymond R. Hayes. Effects of sample size in classifier design. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-11:873–885, 1989.

[28] Stewart Geman, Elie Bienenstock, and René Doursat. Neural networks and the bias/variance dilemma. *Neural Networks*, 4(1):1–58, 1992.

[29] Stephen F. Gull. Bayesian inductive inference and maximum entropy. In Gary L. Ericson and C. Ray Smith, editors, *Maximum Entropy and Bayesian Methods in Science and Engineering 1: Foundations*, volume 1, pages 53–74. Kluwer, Boston, MA, 1988.

[30] Lars Kai Hansen and Peter Salamon. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-12(10):993–1001, 1990.

[31] Thomas Heskes. Bias/variance decompositions for likelihood-based estimators. *Neural Computation*, 10(6):1425–1433, 1998.

[32] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

[33] Robert C. Holte. Very simple classification rules perform well on most commonly used data sets. *Machine Learning*, 11(1):63–91, 1993.

[34] Robert A. Jacobs. Bias/variance analyses of mixtures-of-experts architectures. *Neural Computation*, 9(2):369–383, 1997.

[35] Anil K. Jain, Richard C. Dubes, and Chaur-Chin Chen. Bootstrap techniques for error estimation. *IEEE Transactions on Pattern Analysis and Applications*, PAMI-9(5):628–633, 1998.

[36] Sanjay Jain, Daniel Osherson, James S. Royer, and Arun Sharma. *Systems that Learn: An Introduction to Learning Theory.* MIT Press, Cambridge, MA, second edition, 1999.

[37] Chuanyi Ji and Sheng Ma. Combinations of weak classifiers. *IEEE Transactions on Neural Networks*, 8(1):32–42, 1997.

[38] Gary D. Kendall and Trevor J. Hall. Optimal network construction by minimum description length. *Neural Computation*, 5(2):210–212, 1993.

[39] Josef Kittler. Combining classifiers: A theoretical framework. *Pattern Analysis and Applications*, 1(1):18–27, 1998.

[40] Josef Kittler, Mohamad Hatef, Robert P. W. Duin, and Jiri Matas. On combining classifiers. *IEEE Transactions on Pattern Analysis and Applications*, PAMI-20(3):226–239, 1998.

[41] Andreæi N. Kolmogorov. Three approaches to the quantitative definition of information. *Problems of Information and Transmission*, 1(1):3–11, 1965.

[42] Andreæi N. Kolmogorov and Vladimir A. Uspensky. On the definition of an algorithm. *American Mathematical Society Translations, Series 2*, 29:217–245, 1963.

[43] Eun Bae Kong and Thomas G. Dietterich. Error-correcting output coding corrects bias and variance. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 313–321, San Francisco, CA, 1995. Morgan Kaufmann.

[44] Anders Krogh and Jesper Vedelsby. Neural network ensembles, cross validation, and active learning. In Gerald Tesauro, David S. Touretzky, and Todd K. Leen, editors, *Advances in Neural Information Processing Systems*, volume 7, pages 231–238, Cambridge, MA, 1995. MIT Press.

[45] S. K. Leung-Yan-Cheong and Thomas M. Cover. Some equivalences between Shannon entropy and Kolmogorov complexity. *IEEE Transactions on Information Theory*, 24(3):331–338, 1978.

[46] Ming Li and Paul M. B. Vitányi. Inductive reasoning and Kolmogorov complexity. *Journal of Computer and System Sciences*, 44(2):343–384, 1992.

[47] Ming Li and Paul M. B. Vitányi. *Introduction to Kolmogorov Complexity and Its Applications*. Springer, New York, NY, second edition, 1997.

[48] David J. C. MacKay. Bayesian interpolation. *Neural Computation*, 4(3):415–447, 1992.

[49] David J. C. MacKay. Bayesian model comparison and backprop nets. In John E. Moody, Stephen J. Hanson, and Richard P. Lippmann, editors, *Advances in Neural Information Processing Systems 4*, pages 839–846, San Mateo, CA, 1992. Morgan Kaufmann.

[50] David J. C. MacKay. The evidence framework applied to classification networks. *Neural Computation*, 4(5):698–714, 1992.

[51] David J. C. MacKay. A practical Bayesian framework for backpropagation networks. *Neural Computation*, 4(3):448–472, 1992.

[52] Isaac Newton and Alexander Koyre (Editor). *Isaac Newton's Philosophiae Naturalis Principia Mathematica*. Harvard University Press, Cambridge, MA, third edition, 1972.

[53] Michael Perrone and Leon N Cooper. When networks disagree: Ensemble methods for hybrid neural networks. In Richard J. Mammone, editor, *Artificial Neural Networks for Speech and Vision*, pages 126–142. Chapman & Hall, London, UK, 1993.

[54] Karl Raimund Popper. *Conjectures and Refutations: The Growth of Scientific Knowledge*. Routledge Press, New York, NY, fifth edition, 1992.

[55] Maurice H. Quenouille. Approximate tests of correlation in time series. *Journal of the Royal Statistical Society B*, 11:18–84, 1949.

[56] Maurice H. Quenouille. Notes on bias in estimation. *Biometrika*, 43:353–360, 1956.

[57] Willard V. Quine. *Ontological relativity and other essays*. Columbia University Press, New York, NY, 1969.

[58] Jorma Rissanen. Modelling by shortest data description. *Automatica*, 14(5):465–471, 1978.

[59] Steven Salzberg. On comparing classifiers: Pitfalls to avoid and a recommended approach. *Data Mining and Knowledge Discovery*, 1(3):317–327, 1997.

[60] Cullen Schaffer. A conservation law for generalization performance. In William W. Cohen and Haym Hirsh, editors, *Proceeding of the Eleventh International Conference on Machine Learning*, pages 259–265, San Francisco, CA, 1994. Morgan Kaufmann.

[61] Robert E. Schapire. The strength of weak learnability. *Machine Learning*, 5(2):197–227, 1990.

[62] Robert E. Schapire. Using output codes to boost multiclass learning problems. In *Machine Learning: Proceedings of the Fourteenth International Conference*, pages 313–321, San Mateo, CA, 1997. Morgan Kaufmann.

[63] Gideon Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 6(2):461–464, 1978.

[64] Holm Schwarze and John Hertz. Discontinuous generalization in large committee machines. In Jack D. Cowan, Gerald Tesauro, and Joshua Alspector, editors, *Advances in Neural Information Processing Systems*, volume 6, pages 399–406, San Mateo, CA, 1994. Morgan Kaufmann.

[65] Amanda J. C. Sharkey, editor. *Combining Artificial Neural Nets: Ensemble and Modular Multi-Net Systems*. Springer-Verlag, London, UK, 1999.

[66] Herbert Simon. Theories of bounded rationality. In C. Bartlett McGuire and Roy Radner, editors, *Decision and Organization: A volume in honor of Jacob Marschak*, chapter 8, pages 161–176. North-Holland, Amsterdam, The Netherlands, 1972.

[67] Padhraic Smyth and David Wolpert. Stacked density estimation. In Michael I. Jordan, Michael J. Kearns, and Sara A. Solla, editors, *Advances in Neural Information Processing Systems*, volume 10, pages xxx–xxx, Cambridge, MA, 1998. MIT Press.

[68] Padhraic Smyth and David Wolpert. An evaluation of linearly combining density estimators via stacking. *Machine Learning*, pages xxx–xxx, 1999.

[69] Ray J. Solomonoff. A formal theory of inductive inference. Part I. *Information and Control*, 7(1):1–22, 1964.

[70] Ray J. Solomonoff. A formal theory of inductive inference. Part II. *Information and Control*, 7(2):224–254, 1964.

[71] David G. Stork. Document and character research in the Open Mind Initiative. In *Proceedings of the International Conference on Document Analysis and Recognition (ICDAR99)*, Bangalore, India, September 1999.

[72] David G. Stork. From theory and models to data collection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-xxx(xxx):xxx–xxx, 1999.

[73] David G. Stork. The Open Mind Initiative. *IEEE Expert Systems and their application*, 14(3):19–20, 1999.

[74] Godfried T. Toussaint. Bibliography on estimation of misclassification. *IEEE Transactions on Information Theory*, IT-20(4):472–279, 1974.

[75] Les Valiant. Theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

[76] Šarūnas Raudys and Anil K. Jain. Small sample size effects in statistical pattern recognition: Recommendations for practitioners. *IEEE Transactions on Pattern Analysis and Applications*, PAMI-13(3):252–264, 1991.

[77] Satoshi Watanabe. *Pattern Recognition: Human and Mechanical*. Wiley, New York, NY, 1985.

[78] Ole Winther and Sara A. Solla. Optimal Bayesian online learning. In Kwok-Yee Michael Wong, Irwin King, and Dit-Yan Yeung, editors, *Theoretical Aspects of Neural Computation: A Multidisciplinary Perspective*, pages 61–70. Springer, New York, NY, 1998.

[79] Greg Wolff, Art Owen, and David G. Stork. Empirical error-confidence curves for neural network and gaussian classifiers. *International Journal of Neural Systems*, 7(3):363–371, 1996.

[80] David H. Wolpert, editor. *The Mathematics of Generalization*. Addison-Wesley, Reading, MA, 1995.

[81] David H. Wolpert. On the Bayesian 'Occam Factors' argument for Occam's razor. In Thomas Petsche, Stephen Jose Hanson, and Jude W. Shavlik, editors, *Computational Learning Theory and Natural Learning Systems, Volume III: Selecting good models*, pages 203–224. MIT Press, Cambridge, MA, 1995.

[82] David H. Wolpert. The relationship between PAC, the statistical physics framework, the Bayesian framework, and the VC framework. In David H. Wolpert, editor, *The Mathematics of Generalization*, pages 117–214. Addison-Wesley, Reading, MA, 1995.

[83] Frank J. Wyman, Dean M. Young, and Danny W. Turner. A comparison of asymptotic error rate for the sample linear discriminant function. *Pattern Recognition*, 23(7):775–785, 1990.

[84] Huaiyu Zhu and Richard Rohwer. No free lunch for cross-validation. *Neural Computation*, 8(7):1421–1426, 1996.

Index