ARTICLE Communicated by Steven Nowlan

The Lack of A Priori Distinctions Between Learning Algorithms

David H. Wolpert
The Santa Fe Institute, 1399 Hyde Park Rd., Santa Fe, NM, 87501, USA

This is the first of two papers that use off-training set (OTS) error to investigate the assumption-free relationship between learning algorithms. This first paper discusses the senses in which there are no a priori distinctions between learning algorithms. (The second paper discusses the senses in which there are such distinctions.) In this first paper it is shown, loosely speaking, that for any two algorithms A and B, there are "as many" targets (or priors over targets) for which A has lower expected OTS error than B as vice versa, for loss functions like zero-one loss. In particular, this is true if A is cross-validation and B is "anti-cross-validation" (choose the learning algorithm with largest cross-validation error). This paper ends with a discussion of the implications of these results for computational learning theory. It is shown that one *cannot* say: if empirical misclassification rate is low, the Vapnik–Chervonenkis dimension of your generalizer is small, and the training set is large, then with high probability your OTS error is small. Other implications for "membership queries" algorithms and "punting" algorithms are also discussed.

> "Even after the observation of the frequent conjunction of objects, we have no reason to draw any inference concerning any object beyond those of which we have had experience." David Hume, in *A Treatise of Human Nature*, Book I, part 3, Section 12.

1 Introduction

Much of modern supervised learning theory gives the impression that one can deduce something about the efficacy of a particular learning algorithm (generalizer) without the need for any assumptions about the target input–output relationship one is trying to learn with that algorithm. At most, it would appear, to make such a deduction one has to know something about the training set as well as about the learning algorithm.

Consider for example the following quotes from some well-known papers: "Theoretical studies link the generalization error of a learning

algorithm to the error on the training examples and the capacity of the learning algorithm (independent of concerns about the target)"; "We have given bounds (independent of the target) on the training set size vs. neural net size needed such that valid generalization can be expected"; "If our network can be trained to classify correctly $\ldots 1 - (1 - \gamma)\varepsilon$ of the k training examples, then the probability its [generalization] error is less than ε is at least [a function, independent of the target, of ε, γ, k, and the learning algorithm]"; "There are algorithms that with high probability produce good approximators regardless of the target function .... We do not need to make any assumption about prior probabilities (of targets)"; "To do Bayesian analysis, it is not necessary to work out the prior (over targets)"; "This shows that (the probability distribution of generalization accuracy) gets concentrated at higher and higher accuracy values as more examples are learned (independent of the target)." Similar statements can be found in the "proofs" that various supervised learning communities have offered for Occam's razor (Blumer *et al.* 1987; Berger and Jeffreys 1992; see also Wolpert 1994a, 1995). There even exists a field ("agnostic learning," Kearns *et al.* 1992) whose expressed purpose is to create learning algorithms that are assuredly effective even in the absence of assumptions about the target.

Frequently the authors of these kinds of quotes understand that there are subtleties and caveats behind them. But the quotes taken at face value raise an intriguing question: can one actually get something for nothing in supervised learning? Can one get useful, caveat-free theoretical results that link the training set and the learning algorithm to generalization error, without making assumptions concerning the target? More generally, are there useful practical techniques that require no such assumptions? As a potential example of such a technique, note that people usually use cross-validation without making any assumptions about the underlying target, as though the technique were universally applicable.

This is the first of two papers that present an initial investigation of this issue. These papers can be viewed as an analysis of the mathematical "skeleton" of supervised learning, before the "flesh" of particular priors over targets and similar problem-specific distributions is introduced. It should be emphasized that the work in these papers is very preliminary; even the "skeleton" of supervised learning is extremely rich and detailed. Much remains to be done.

The primary mathematical tool used in these papers is off-training set (OTS) generalization error, i.e., generalization error for test sets that contain no overlap with the training set. (In the conventional measure of generalization error such overlap is allowed.) Section 2 of this first paper explains why such a measure of error is of interest, and in particular emphasizes that it is equivalent to (more conventional) IID error in many scenarios of interest. Those who already accept that OTS error is of interest can skip this section.

Section 3 presents the mathematical formalism used in this paper.

Section 4 presents the "no free lunch" (NFL) theorems (phrase due to D. Haussler). Some of those theorems show, loosely speaking, that for any two algorithms A and B, there are "as many" targets for which algorithm A has lower expected OTS error than algorithm B as vice versa (whether one averages over training sets or not). In particular, such equivalence holds even if one of the algorithms is random guessing; there are "as many" targets for which any particular learning algorithm gets confused by the data and performs *worse than random* as for which it performs better. As another example of the NFL theorems, it is shown explicitly that A is equivalent to B when B is an algorithm that chooses between two hypotheses based on which *dis*agrees more with the training set, and A is an algorithm that chooses based on which agrees more with the training set. Other NFL theorems are also derived, showing, for example, that there are as many priors over targets in which A beats B (i.e., has lower expected error than B) as vice versa. In all this, the quotes presented at the beginning of this section are misleading at best.

Next a set of simple examples is presented illustrating the theorems in scenarios in which their applicability is somewhat counterintuitive. In particular, a brief discussion is presented of the fact that there are as many targets for which it is preferable to choose between two learning algorithms based on which has *larger* cross-validation error ("anti-cross-validation") as based on which has smaller cross-validation error.

This section also contains the subsection "Extensions for nonuniform averaging" that extends the NFL results beyond uniform averages; as that subsection shows, one can, for example, consider only priors over targets that are highly structured, and it is still often true that all algorithms are equal. Also in this section is the subsection "On uniform averaging," which provides the intellectual context for the analyses that result in the NFL theorems.

Section 5 discusses the NFL theorem's implications for and relationship with computational learning theory. It starts with a discussion of empirical error and OTS error. This discussion makes clear that one must be very careful in trying to interpret uniform convergence (VC) results. In particular, it makes clear that one *cannot* say: if the observed empirical misclassification rate is low, the VC dimension of your generalizer is small, and the training set is large, then with high probability your OTS error is small. After this, the implications of the NFL results for active learning, and for "membership queries" algorithms and "punting" algorithms (those that may refuse to make a guess), are discussed.

Small and simple proofs of claims made in the text of this first paper are collected in Appendix C.

Paper one concentrates on relative sizes of sets of targets and the associated senses in which all algorithms are a priori equivalent. In contrast, paper two concentrates on other ways to compare algorithms. Some of these alternative comparisons reveal no distinctions between algorithms, just like the comparisons in paper one. However some of the other al-

ternative comparisons result in a priori distinctions between algorithms. In particular, it is pointed out in paper two that the equivalence of average OTS error between cross-validation and anti-cross-validation does not mean they have equivalent "head-to-head minimax" properties, and that algorithms can differ in those properties. Indeed, it may be that cross-validation has better head-to-head minimax properties than anti-cross-validation, and therefore can be a priori justified in that sense.

Of course, the analysis of paper one does not rule out the possibility that there are targets for which a particular learning algorithm works well compared to some other one. To address the nontrivial aspects of this issue, paper two discusses the case where one averages over hypotheses rather than targets. The results of such analyses hold for all possible priors, since they hold for all (fixed) targets. This allows them to be used to prove, as a particular example, that cross-validation cannot be justified as a Bayesian procedure, i.e., there is no prior over targets for which, *without regard for the learning algorithms in question*, one can conclude that one should choose between those algorithms based on minimal rather than (for example) maximal cross-validation error. In addition, it is noted that for a very natural restriction of the class of learning algorithms, one can distinguish between using minimal rather than maximal cross-validation error—and the result is that one should use maximal error(!).

All of the analysis up to this point assumes the loss function is in the same class as the zero-one loss function (which is assumed in almost all of computational learning theory). Paper two goes on to discuss other loss functions. In particular, the quadratic loss function modifies the preceding results considerably; for that loss function, there *are* algorithms that are a priori superior to other algorithms. However, it is shown in paper two that no algorithm is superior to its "randomized" version, in which the set of potential fits to training sets is held fixed, but which fit is associated with which training set changes. In this sense one cannot a priori justify any particular learning algorithm, even for a quadratic loss function.

Finally, paper two ends with a brief overview of some open issues and discusses future work.

It cannot be emphasized enough that no claim is being made in this first paper that all algorithms are equivalent *in practice*, in the real world. In particular, no claim is being made that one should not use cross-validation in the real world. (I have done so myself many times in the past and intend to do so again in the future.) The sole concern of this paper is what can(not) be formally inferred about the utility of various learning algorithms if one makes no assumptions concerning targets.

The work in these papers builds upon the analysis in Wolpert (1992, 1993). Some aspects of that early analysis are nicely synopsized in Schaffer (*1993*, *1994*). Schaffer (1994) also contains an interesting discussion of the implications of the NFL theorems for real world learning, as does

Murphy and Pazzani (1994). See also Wolpert and Macready (1995) for related work in the field of combinatorial optimization.

The major extensions beyond this previous work that is contained in these two papers are (1) many more issues are analyzed (e.g., essentially all of paper two was not touched upon in the earlier work); and (2) many fewer restrictions are made (e.g., losses other than zero-one are considered, arbitrary kinds of noise are allowed, both hypotheses and targets are arbitrary probability distributions rather than single-valued functions from inputs to outputs, etc.).

2 Off-Training-Set Error

Many introductory supervised learning texts take the view that "the overall objective ... is to learn from samples and to generalize to new, *as yet unseen cases*" (italics mine—see Weiss and Kulikowski 1991, for example). Similarly, in supervised learning it is common practice to try to avoid fitting the training set exactly, to try to avoid "overtraining." One of the major rationales given for this is that if one overtrains, "the resulting (system) is unlikely to classify *additional points* (in the input space) correctly" (italics mine—see Dietterich 1990). As another example, in Blumer *et al.* (1987), we read that "the real value of a scientific explanation lies not in its ability to explain (what one has already seen), but in predicting events that have yet to (be seen)." As a final example, in Mitchell and Blum (1994) we read that "(in Machine Learning we wish to know whether) any hypothesis found to approximate the target function well over a sufficiently large set of training examples will also approximate the target function well over *other unobserved* examples."

This language makes clear that OTS behavior is a central concern of supervised learning, even though little theoretical work has been devoted to it to date. Some of the reasons for such concern are as follows.

1. In the low-noise (for outputs) regime, optimal behavior *on the training set* is trivially determined by lookup table memorization. Of course, this has nothing to do with behavior off of the training set; so in this regime, it is only such OTS behavior that is of interest.
2. In particular, in that low-noise regime, if one uses a memorizing learning algorithm, then for test sets overlapping with training sets the upper limit of possible test set error values shrinks as the training set grows. If one does not correct for this when comparing behavior for different sizes of the training set (as when investigating learning curves), one is comparing apples and oranges. In that low-noise regime, correcting for this effect by renormalizing the range of possible error values is equivalent to requiring that test sets and training sets be distinct, i.e., is equivalent to using OTS error (see Wolpert 1994a).

3. In artificial intelligence—one of the primary fields concerned with supervised learning—the emphasis is often exclusively on generalizing to as yet unseen examples.
4. In the real world, very often the process generating the training set is not the same as that governing testing. In such scenarios, the usual justification for testing with the same process that generated the training set (and with it the possibility that test sets overlap with training sets) does not apply.

One example of such a difference between testing and training is "active" or "query-based" or "membership-based" learning. In that kind of learning the learner chooses, perhaps dynamically, where in the input space the training set elements are to be. However, conventionally, there is no such control over the test set. So testing and training are governed by different processes.

As another example, say we wish to learn tertiary protein structure from primary structure and then use that to aid drug design. We *already know* what tertiary structure corresponds to the primary structures in the training set. So we will never have those structures in the "test set" (i.e., in the set of nucleotide sequences whose tertiary structure we wish to infer to aid the drug design process). We will *only* be interested in OTS error.

5. Distinguishing the regime where test examples coincide with the training set from the one where there is no overlap amounts to splitting supervised learning along its natural "cleavage plane." Since behavior can be radically different in the two regimes, it is hard to see why one wouldn't want to distinguish them.
6. When the training set is much smaller than the full input space, the probability that a randomly chosen test set input value coincides with the training set is vanishingly small. So in such situations one expects the value of the OTS error to be well-approximated by the value of the conventional IID (independent identically distributed) error, an error that allows overlap between test sets and training sets.

   One might suppose that in such a small training set regime there is no aspect of OTS error not addressable by instead calculating IID error. This is wrong though, as the following several points illustrate.
7. First, even if OTS error is well approximated by IID error, it does not follow that quantities like the "derivatives" of the errors are close to one another. In particular, it does not follow that the sign of the slope of the learning curve—often an object of major interest—is the same for both errors over some region of interest.

As an example, in Wolpert *et al.* (1995), it is shown that the expected OTS misclassification rate can *increase* with training set size, even if one

averages both over training sets and targets, and even if one uses the Bayes-optimal learning algorithm. In contrast, it is also shown there that under those same conditions, the expected IID misclassification rate is strictly nonincreasing as a function of training set size for the Bayes-optimal learning algorithm (see also the discussion in Wolpert 1994a concerning the statistical physics supervised learning formalism).

8. Second, although it is usually true that a probability distribution over IID error will well-approximate the corresponding distribution over OTS error, distributions *conditioned* on IID error can differ drastically from distributions conditioned on OTS error. This can be very important in understanding the results of computational learning theory.

As an example of such a difference, let s be the empirical misclassification rate between a hypothesis and the target over the training set (i.e., the average number of disagreements over the training set), m the size of the training set, c'_{IID} the misclassification rate over all of the input space (the IID zero-one loss generalization error), and c'_{OTS} the misclassification rate over that part of the input space lying outside of the training set. (These terms are formally defined in the next section and at the beginning of Section 5.) Assume a uniform sampling distribution over the input space, a uniform prior over target input–output relationships, and a noise-free IID likelihood governing the training set generation. Then $P(s \mid c'_{\text{IID}}, m)$, the probability of getting empirical misclassification rate s given global misclassification rate c'_{IID}, averaged over all training sets of size m, is just the binomial distribution $(c'_{\text{IID}})^{sm}(1 - c'_{\text{IID}})^{(m-sm)}C^m_{sm}$, where $C^a_b \equiv a!/[b!(a-b)!]$ (s can be viewed as the percentage of heads in m flips of a coin with bias c'_{IID} toward heads).

On the other hand, $P(s \mid c'_{\text{OTS}}, m)$, the probability of getting empirical misclassification rate s given off-training sets misclassification rate c'_{OTS}, averaged over all training sets of size m, is independent of c'_{OTS}. (This is proven in Section 5 below.) So the dependence of the empirical misclassification rate on the global misclassification rate depends crucially on whether it is OTS or IID "global misclassification rate."

9. Third, often it is more straightforward to calculate a certain quantity for OTS rather than IID error. In such cases, even if one's ultimate interest is IID error, it makes sense to instead calculate OTS error (assuming one is in a regime where OTS error well-approximates IID error).

As an example, OTS error results presented in Section 5 mean that when the training set is much smaller than the full input space, $P(c'_{\text{IID}} \mid s, m)$ is (arbitrarily close to) independent of s, if the prior over target input–output relationships is uniform. This holds despite VC results saying that independent of the prior, it is highly unlikely for c'_{IID} and s to differ significantly. (This may seem paradoxical at first. See the discussion in Section 5 for the "resolution.")

The formal identity (in the appropriate limit) between a probability distribution over an OTS error and one over an IID error is established at the end of Appendix B.

None of the foregoing means that the conventional IID error measure is "wrong." No claim is being made that one "should not" test with the same process that generated the training set. Rather the claim is simply that OTS testing is an issue of major importance. In that it gives no credit for memorization, it is also the natural way to investigate whether one can make assumption-free statements concerning an algorithm's generalization (!) ability.

3 The Extended Bayesian Formalism

These papers use the extended Bayesian formalism (EBF—Wolpert 1992, 1994a; Wolpert *et al.* 1995). In the current context, the EBF is just conventional probability theory, applied to the case where one has a different random variable for the hypothesis output by the learning algorithm and for the target relationship. It is this crucial distinction that separates the EBF from conventional Bayesian analysis, and that allows the EBF (unlike conventional Bayesian analysis) to subsume all other major mathematical treatments of supervised learning like computational learning theory, sampling theory statistics, etc. (see Wolpert 1994a).

This section presents a synopsis of the EBF. Points (2), (8), (14), and (15) below can be skipped in a first reading. A quick reference of this section's synopsis can be found in Table 1.

Readers unsure of any aspects of this synopsis, and in particular unsure of any of the formal basis of the EBF or justifications for any of its (sometimes implicit) assumptions, are directed to the detailed exposition of the EBF in Appendix A.

3.1 Overview.

1. The input and output spaces are $\mathbf{X}$ and $\mathbf{Y}$, respectively. They contain n and r elements, respectively. A generic element of $\mathbf{X}$ is indicated by x, and a generic element of $\mathbf{Y}$ is indicated by y.
2. Random variables are indicated using capital letters. Associated instantiations of a random variable are indicated using lower case letters. Note though that some quantities (e.g., the space $\mathbf{X}$) are neither random variables nor instantiations of random variables, and therefore their written case carries no significance.

   Only rarely will it be necessary to refer to a random variable rather than an instantiation of it. In accord with standard statistics notation, "$E(A \mid b)$" will be used to mean the expectation value of A given $B = b$, i.e., to mean $\int da\, a P(a \mid b)$. (Sums replace integrals if appropriate.)

Table 1: Summary of the Terms in the EBF

The sets **X** and **Y**, of sizes n and r	The input and output space, respectively.
The set d, of m **X**-**Y** pairs	The training set.
The **X**-conditioned distribution over **Y**, f	The target, used to generate test sets.
The **X**-conditioned distribution over **Y**, h	The hypothesis, used to guess for test sets.
The real number c	The cost.
The **X**-value q	The test set point.
The **Y**-value y_F	The sample of the target f at point q.
The **Y**-value y_H	The sample of the hypothesis h at point q.
$P(h \mid d)$	The learning algorithm.
$P(f \mid d)$	The posterior.
$P(d \mid f)$	The likelihood.
$P(f)$	The prior.

If $c = L(y_F, y_H)$, $L(\cdot, \cdot)$ is the *"loss function"*

L is *"homogeneous"* if $\sum_{y_F} \delta[c, L(y_H, y_F)]$ is independent of y_H

If we restrict attention to fs given by a fixed noise process superimposed on an underlying single-valued funtion from **X** to **Y**, ϕ, and if $\sum_{\phi} P(y_F \mid q, \phi)$ is independent of y_F, we have "homogeneous" noise

3. The primary random variables are the hypothesis **X**–**Y** relationship output by the learning algorithm (indicated by H), the target (i.e., "true") **X**–**Y** relationship (F), the training set (D), and the real world cost (C).

These variables are related to one another through other random variables representing the (test set) input space value (Q), and the associated target and hypothesis **Y**-values, Y_F and Y_H, respectively (with instantiations y_F and y_H, respectively).

This completes the list of random variables.

As an example of the relationship between these random variables and supervised learning, f, a particular instantiation of a target, could refer to a "teacher" neural net together with superimposed noise. This noise-corrupted neural net generates the training set d. The hypothesis h on the other hand could be the neural net made by one's "student" algorithm after training on d. Then q would be an input element of the test set, y_F and y_H associated samples of the outputs of the two neural

nets for that element (the sampling of y_F including the effects of the superimposed noise), and c the resultant "cost" [e.g., c could be $(y_F - y_H)^2$].

3.2 Training Sets and Targets.

4. m is the number of elements in the (ordered) training set d. $\{d_{\mathbf{X}}(i), d_{\mathbf{Y}}(i)\}$ is the set of m input and output values in d. m' is the number of distinct values in $d_{\mathbf{X}}$.
5. Targets f are always assumed to be of the form of **X**-conditioned distributions over **Y**, indicated by the real-valued function $f(x \in \mathbf{X}, y \in \mathbf{Y})$ [i.e., $P(y_F \mid f, q) = f(q, y_F)$]. Equivalently, where S_r is defined as the r-dimensional unit simplex, targets can be viewed as mappings $f : \mathbf{X} \to S_r$.

Any restrictions on f are imposed by $P(f, h, d, c)$, and in particular by its marginalization, $P(f)$. Note that any output noise process is automatically reflected in $P(y_F \mid f, q)$. Note also that the definition $P(y_F \mid f, q) = f(q, y_F)$ only directly refers to the generation of test set elements; in general, training set elements can be generated from targets in a different manner.

6. The "likelihood" is $P(d \mid f)$. It says how d was generated from f. It is "vertical" if $P(d \mid f)$ is independent of the values $f(x, y_F)$ for those $x \notin d_{\mathbf{X}}$. As an example, the conventional IID likelihood is

$$P(d \mid f) = \prod_{i=1}^{m} \pi[d_{\mathbf{X}}(i)]\, f[d_{\mathbf{X}}(i), d_{\mathbf{Y}}(i)] \tag{3.1}$$

[where $\pi(x)$ is the "sampling distribution"]. In other words, under this likelihood d is created by repeatedly and independently choosing an input value $d_{\mathbf{X}}(i)$ by sampling $\pi(\cdot)$, and then choosing an associated output value by sampling $f[d_{\mathbf{X}}(i), \cdot]$, the same distribution used to generate test set outputs. This likelihood is vertical.

As another example, if there is noise in generating training set **X** values but none for test set **X** values, then we usually do not have a vertical $P(d \mid f)$. (This is because, formally speaking, f directly governs the generation of test sets, not training sets; see Appendix A.)

7. The "posterior" usually means $P(f \mid d)$, and the "prior" usually means $P(f)$.
8. It will be convenient at times to restrict attention to fs that are constructed by adding noise to a single-valued function from **X** to **Y**, ϕ. For a fixed noise process, such fs are indexed by the underlying ϕ.

The noise process is "homogeneous" if the sum over all ϕ of $P(y_F \mid q, \phi)$ is independent of y_F. An example of a homogeneous noise process is classification noise that with probability p replaces $\phi(q)$ with some other value in **Y**, where that "other value in **Y**" is chosen uniformly and randomly.

3.3 The Learning Algorithm.

9. Hypotheses h are always assumed to be of the form of **X**-conditioned distributions over **Y**, indicated by the real-valued function $h(x \in \mathbf{X}, y \in \mathbf{Y})$ [i.e., $P(y_H \mid h, q) = h(q, y_H)$]. Equivalently, where S_r is defined as the r-dimensional unit simplex, hypotheses can be viewed as mappings $h : \mathbf{X} \to S_r$.

Any restrictions on h are imposed by $P(f, h, d, c)$. Here and throughout, a "single-valued" distribution is one that, for a given x, is a delta function about some y. Such a distribution is a single-valued function from **X** to **Y**. As an example, if one is using a neural net as one's regression through the training set, usually the (neural net) h is single-valued. On the other hand, when one is performing probabilistic classification (as in softmax), h is not single-valued.

10. Any (!) learning algorithm (aka "generalizer") is given by $P(h \mid d)$, although writing down a learning algorithm's $P(h \mid d)$ explicitly is often quite difficult. A learning algorithm is "deterministic" if the same d always gives the same h. Backprop with a random initial weight is not deterministic. Nearest neighbor is.

Note that since d is ordered, "on-line" learning algorithms are subsumed as a special case.

11. The learning algorithm only sees the training set d, and in particular does not directly see the target. So $P(h \mid f, d) = P(h \mid d)$, which means that $P(h, f \mid d) = P(h \mid d) \times P(f \mid d)$, and therefore $P(f \mid h, d) = P(h, f \mid d)/P(h \mid d) = P(f \mid d)$.

3.4 The Cost and "Generalization Error".

12. For the purposes of this paper, the cost c is associated with a particular y_H and y_F, and is given by a loss function $L(y_H, y_F)$. As an example, in regression, often we have "quadratic loss": $L(y_H, y_F) = (y_H - y_F)^2$.

$L(\cdot, \cdot)$ is "homogeneous" if the sum over y_F of $\delta[c, L(y_H, y_F)]$ is some function $\Lambda(c)$, independent of y_H (δ here being the Kronecker delta function). As an example, the "zero-one" loss traditional in computational learning theory [$L(a, b) = 1$ if $a \neq b$, 0 otherwise] is homogeneous.

13. In the case of "IID error" (the conventional error measure), $P(q \mid d) = \pi(q)$ (so test set inputs are chosen according to the same distribution that determines training set inputs). In the case of OTS error, $P(q \mid d) = [\delta(q \notin d_{\mathbf{X}})\pi(q)]/[\sum_q \delta(q \notin d_{\mathbf{X}})\pi(q)]$, where $\delta(z) \equiv 1$ if z is true, 0 otherwise.

Subscripts OTS or IID on c correspond to using those respective kinds of error.

14. The "generalization error function" used in much of supervised learning is given by $c' \equiv E(C \mid f, h, d)$. (Subscripts OTS or IID on c' correspond to using those respective ways to generate q.) It is the average over all q of the cost c, for a given target f, hypothesis h, and training set d.

In general, probability distributions over c' do not by themselves determine those over c or vice versa, i.e., there is not an injection between such distributions. However, the results in this paper in general hold for both c and c', although they will be presented only for c. In addition, especially when relating results in this paper to theorems in the literature, sometimes results for c' will implicitly be meant even when the text still refers to c. (The context will make this clear.)

15. When the size of $\mathbf{X}$, n, is much greater than the size of the training set, m, probability distributions over c'_{IID} and distributions over c'_{OTS} become identical. (Although, as mentioned in the previous section, distributions conditioned on c'_{IID} can be drastically different from those conditioned on c'_{OTS}.) This is established formally in Appendix B.

4 The No-Free-Lunch Theorems

In Wolpert (1992) it is shown that $P(c \mid d) = \int df\, dh\, P(h \mid d) P(f \mid d) M_{c,d}(f, h)$, where so long as the loss function is symmetric in its arguments, $M_{c,d}(\cdot, \cdot)$ is symmetric in its arguments. (See point (11) of the previous section.) In other words, for the most common kinds of loss functions (zero-one, quadratic, etc.), the probability of a particular cost is determined by an inner product between your learning algorithm and the posterior probability. [f and h being the component labels of the d-indexed infinite-dimensional vectors $P(f \mid d)$ and $P(h \mid d)$, respectively.] Metaphorically speaking, how "aligned" you (the learning algorithm) are with the universe (the posterior) determines how well you will generalize.

The question arises though of how much can be said concerning a particular learning algorithm's generalization behavior without specifying the posterior (which usually means without specifying the prior). More precisely, the goal is to address the issue of how F_1, the set of targets f for which algorithm A outperforms algorithm B, compares to F_2, the set of targets f for which the reverse is true. To analyze this issue, the simple trick is used of comparing the average over f of f-conditioned probability distributions for algorithm A to the same average for algorithm B. The relationship between those averages is then used to compare F_1 to F_2.

Evaluating such f-averages results in a set of NFL theorems. In this section, first I derive the NFL theorems for the case where the target f need not be single-valued. In this case, the theorems say that uniformly averaged over all f, all learning algorithms are identical. The implications

of this for how F_1 compares to F_2 are discussed after the derivation of the theorems.

When the target f is not single-valued, it is a (countable) set of real numbers (one for each possible x–y pair). Accordingly, any $P(f)$ is a probability density function in a multidimensional space. That makes integrating over all $P(f)$s a subtle mathematical exercise. However in the function+noise scenario, for a fixed noise process, "f" is indexed by a single-valued function ϕ. Since there are a countable number of ϕs, any $P(\phi)$ is a countable set of real numbers, and it is straightforward to integrate over all $P(\phi)$. Doing so gives some more NFL theorems, where one uniformly averages over all priors rather than just over all targets. These additional theorems are presented after those involving averages over all targets f.

After deriving these theorems, I present some examples of them, designed to highlight their counterintuitive aspects. I also present a general discussion of the significance of the theorems, and in particular of the uniform averaging that goes into deriving them.

Here and throughout this paper, when discussing non-single-valued fs, "$A(f)$ uniformly averaged over all targets f" means $\int df\, A(f) / \int df\, 1$. Note that these integrals are implicitly restricted to those f that constitute **X**-conditioned distributions over **Y**, i.e., to the appropriate product space of unit-simplices. (The details will not matter, because integrals will almost never need to be evaluated. But formally, integrals over targets f are over a full r^n-dimensional Euclidean space, with a product of Dirac delta functions and Heaviside functions inside the integrand enforcing the restriction to the Cartesian product of simplices.)

Similar meanings for "uniformly averaged" are assumed if we are talking about averaging over other quantities, like $P(\phi)$.

4.1 Averaging over All Target Distributions f. We start with the following simple lemma, that recurs frequently in the subsequent analysis. Its proof is in Appendix C.

Lemma 1. $P(c \mid f, d) = \sum_{y_H, y_F, q} \delta[c, L(y_H, y_F)]\, P(y_H \mid q, d) P(y_F \mid q, f) P(q \mid d)$

Consider now the "(uniform) random learning algorithm": for any test set element not in the training set, guess the output randomly (independently of the training set d), according to a uniform distribution. (With certain extra stipulations concerning behavior for test set questions $q \in d_{\mathbf{X}}$, this is a version of the Gibbs learning algorithm.) An immediate corollary of Lemma (1), proven in Appendix C, is that for this algorithm, for a symmetric homogeneous loss function, $P(c \mid d) = \Lambda(c)/r$ for all training sets d. Similarly, for all priors over targets f, indicated by α, both $P(c \mid m, \alpha)$ and $P(c \mid d, \alpha)$ equal $\Lambda(c)/r$, for this random learning algorithm.

This simple kind of reasoning suffices to get "NFL" results for the random algorithm, even without invoking a vertical likelihood. However, more is needed for scenarios concerning other algorithms, scenarios in which there is "randomness," but it concerns targets rather than hypotheses. This is because we are interested in probability distributions conditioned on target-based quantities (f, α, etc.), so results for when there is randomness in hypothesis-based quantities do not immediately carry over to results for randomness in target-based quantities.

To analyze these alternative scenarios, we start with the following simple implication of Lemma (1) (see Appendix C):

The uniform average over all targets f of $P(c \mid f, d)$ equals

$$(1/r) \sum_{y_H, y_F, q} \delta[c, L(y_H, y_F)] \, P(y_H \mid q, d) \, P(q \mid d)$$

Recalling the definition of homogenous loss L, we have now proven the following:

Theorem 1. *For homogeneous loss L, the uniform average over all f of $P(c \mid f, d)$ equals $\Lambda(c)/r$.*

Note that this f-average is independent of the learning algorithm. So Theorem (1) constitutes an NFL theorem for distributions conditioned on targets f and training sets d; it says that uniformly averaged over all f, such distributions are independent of the learning algorithm. Note that this result holds independent of the sampling distribution, the training set, or the likelihood.

As an example of Theorem (1), for the $\Lambda(c)$ of zero-one loss, we get the f-average of $E(C \mid f, d) = r - 1/r$. More generally, for an even broader set of loss functions L than homogeneous Ls, the sum over target outputs y_F of $L(y_H, y_F)$ is independent of the hypothesis output, y_H. For such Ls we get generalizer-independence for the uniform average over targets f of $E(C \mid f, d)$, even if we do not have such independence for the uniform average of $P(c \mid f, d)$.

Note that Theorem (1) does not rely on having q lie outside of d_X; it holds even for IID error. In addition, since both f and d are fixed in the conditional probability in Theorem (1), any statistical coupling between f and d is ignored in that theorem. For these kinds of reasons, Theorem (1) is not too interesting by itself. The main use of it is to derive other results, results that rely on using OTS error and that are affected by the coupling of targets f and training sets d. As the first of these, I will show how to use Theorem (1) to evaluate the uniform f-average of $P(c \mid f, m)$ for OTS error.

In evaluating the uniform f-average of $P(c \mid f, m)$, not all fs contribute the same amount to the answer. That is because

$$P(c \mid f, m) = \sum_d P(c \mid f, d) \, P(d \mid f)$$

and so long as the likelihood $P(d \mid f)$ is not uniform over f, we cannot just pull the outside f-average through to use Theorem (1) to reduce the $P(c \mid f, d)$ to $\Lambda(c)/r$. This might lead one to suspect that if the learning algorithm is "biased" toward the targets f contributing the most to the uniform f-average of $P(c \mid f, m)$, then the average would be weighted toward (or away from) low values of cost, c. However this is wrong; it turns out that the uniform f-average of $P(c \mid f, m)$ is independent of the learning algorithm, if one restricts oneself to OTS error.

In fact, assume that we have any $P(q \mid d)$ such that $P(q \in d_X \mid d) = 0$ [in particular, $P(q \mid d)$ need not be the OTS $P(q \mid d)$ discussed above]. For such a scenario, for a vertical likelihood [i.e., a $P(d \mid f)$ that is independent of the values of $f(x \notin d_X, \cdot)$], we get the following result (see Appendix C):

Theorem 2. *For OTS error, a vertical $P(d \mid f)$, and a homogeneous loss L, the uniform average over all targets f of $P(c \mid f, m) = \Lambda(c)/r$.*

Again, this holds for any learning algorithm, and any sampling distribution. Note that this result means in particular that the "weight" of fs on which one's algorithm performs worse than the random algorithm equals the weight for which it performs better. In other words, one can just as readily have a target for which one's algorithm has *worse than random guessing* as one in which it performs better than random. The pitfall we wish to avoid in supervised learning is not simply that our algorithm performs as poorly as random guessing, but rather that our algorithm performs worse than randomly!

Using similar reasoning to that used to prove Theorem (2), we can derive the following theorem concerning the distribution of interest in conventional Bayesian analysis, $P(c \mid d)$:

Theorem 3. *For OTS error, a vertical $P(d \mid f)$, uniform $P(f)$, and a homogeneous loss L, $P(c \mid d) = \Lambda(c)/r$.*

The reader should be wary of equating the underlying logic behind a target-averaging NFL theorem [e.g., Theorem (2)] with that behind a uniform-prior NFL theorem [e.g., Theorem (3)]. In particular, there are scenarios [i.e., conditioning events in the conditional distribution "$P(c \mid \ldots)$"] in which one of these kinds of NFL theorem holds but not the other. See the discussion surrounding Theorem (9) below for an example.

As an immediate corollary of Theorem (3), we have the following.

Corollary 1. *For OTS error, a vertical $P(d \mid f)$, uniform $P(f)$, and a homogeneous loss L, $P(c \mid m) = \Lambda(c)/r$.*

As an aside, so long as $L(a, b) = L(b, a)$ for all pairs a and b, the mathematics of the EBF is symmetric under interchange of h and f. [In particular, for any loss L, it is both true that $P(f \mid h, d) = P(f \mid d)$, and

that $P(h \mid f, d) = P(h \mid d)$.] Accordingly, all of the NFL theorems have analogues where the hypothesis h rather than the target f is fixed and then uniformly averaged over. So for example, for OTS error, homogeneous $L(\cdot, \cdot)$, and a generalizer such that $P(d \mid h)$ is independent of $h(x \notin d_{\mathbf{X}})$, the uniform average over h of $P(c \mid h, m) = \Lambda(c)/r$. [For such a non-deterministic generalizer, assuming $h_1(x) = h_2(x)$ for all $x \in d_{\mathbf{X}}$, the probability that the training set used to produce the hypothesis was d is the same, whether that produced hypothesis is h_1 or h_2.] Such results say that averaged over all hs the algorithm might produce, all posteriors over targets (and therefore all priors) lead to the same probability of cost, under the specified conditions.

4.2 Averaging over All Functions ϕ. Now consider the scenario where only those targets f are allowed that can be viewed as single-valued functions ϕ from $\mathbf{X}$ to $\mathbf{Y}$ with noise superimposed (see Section 3). To analyze such a scenario, I will no longer consider uniform averages involving f directly, but rather uniform averages involving ϕ. Accordingly, such averages are now sums rather than integrals. (For reasons of space, only here in this subsection will I explicitly consider the case of fs that are single-valued ϕs with noise superimposed.)

In this new scenario, Lemma (1) still holds, with f replaced by ϕ. However now we cannot simply set the uniform ϕ-average of $P(y_F \mid q, \phi)$ to $1/r$, in analogy to the reasoning implicitly used above [see the proof of the "implication of Lemma (1)" in Appendix C]. To give an extreme example, if the test set noise process is highly skewed and sends all $\phi(q)$ to some fixed value y_1, then the ϕ-average is 1 for $y_F = y_1$, 0 otherwise. Intuitively, if the noise process always results in the test value y_1, then we *can* make a priori distinctions between learning algorithms; an algorithm that always guesses y_1 outside of $d_{\mathbf{X}}$ will beat one that does not.

So for simplicity restrict attention to those noise processes for which the uniform ϕ-average of $P(y_F \mid q, \phi)$ is independent of the target output value y_F. Recall that such a (test set) noise process is called "homogeneous." So following along with our previous argument (recounted in Appendix C), if we sum our ϕ-average of $P(y_F \mid q, \phi)$ over all y_F, then by pulling the sum over y_F inside the average over ϕ, we see that the sum must equal 1. [Again, see the proof of the "implication of Lemma (1)."] Accordingly, the ϕ-average equals $1/r$. So we have the following analog of Theorem (1):

Theorem 4. *For homogeneous loss L and a homogeneous test-set noise process, the uniform average over all single-valued target functions ϕ of $P(c \mid \phi, d)$ equals $\Lambda(c)/r$.*

Note that the noise process involved in generating the training set is irrelevant to this result. (Recall that "homogeneous noise" refers to y_F and y_H, and that y_F and y_H are $\mathbf{Y}$ values for the test process, not the

training process.) This is also true for the results presented below. So in particular, all these results hold for any noise in the generation of the training set, so long as our error measure is concerned with whether or not h equals the (homogeneous noise corrupted) sample of the underlying ϕ at the test point q. (Note, in particular, that such a measure is always appropriate for noise-free—and therefore trivially homogeneous—test set generation).

We can proceed from Theorem (4) to get a result for $P(c \mid f, m)$ in the exact same manner as Theorem (1) gave Theorem (2).

Theorem 5. *For OTS error, a vertical $P(d \mid \phi)$, homogeneous loss L, and a homogeneous test-set noise process, the uniform average over all single-valued target functions ϕ of $P(c \mid \phi, m)$ equals $\Lambda(c)/r$.*

Just as the logic behind Theorem (2) also resulted in Theorem (3), so we can use the logic behind Theorem (5) to derive the following.

Theorem 6. *For OTS error, a vertical $P(d \mid \phi)$, homogeneous loss L, uniform $P(\phi)$, and a homogeneous test-set noise process, $P(c \mid d)$ equals $\Lambda(c)/r$.*

Just as Theorem (3) resulted in Corollary (1), so Theorem (6) establishes the following.

Corollary 2. *For OTS error, vertical $P(d \mid \phi)$, homogeneous loss L, a homogeneous test-set noise process, and uniform $P(\phi)$, $P(c \mid m)$ equals $\Lambda(c)/r$.*

We are now in a position to extend the NFL theorems to the case where neither the prior nor the target is specified in the conditioning event of our distribution of interest, and the prior need not be uniform. For such a case, the NFL results concern uniformly averaging over priors $P(\phi)$ rather than over target functions ϕ.

Since there are r^n possible single-valued ϕ, $P(\phi)$ is an r^n-dimensional real-valued vector lying on the unit simplex. Indicate that vector as α, and one of its components [i.e., $P(\phi)$ for one ϕ] as α_ϕ. [More formally, α is a hyperparameter: $P(\phi \mid \alpha) \equiv \alpha_\phi$.] So the uniform average over all α of $P(c \mid m, \alpha)$ is (proportional to) $\int d\alpha P(c \mid m, \alpha) = \int d\alpha [\sum_\phi P(\phi \mid \alpha) P(c \mid m, \alpha, \phi)]$, where the integral is restricted to the r^n-dimensional simplex. [α is restricted to lie on that simplex, since $\sum_\phi P(\phi \mid \alpha) = \sum_\phi \alpha_\phi = 1$.] It is now straightforward to use Theorem (5) to establish the following result (see Appendix C):

Theorem 7. *Assume OTS error, a vertical $P(d \mid \phi)$, homogeneous loss L, and a homogeneous test-set noise process. Let α index the priors $P(\phi)$. Then the uniform average over all α of $P(c \mid m, \alpha)$ equals $\Lambda(c)/r$.*

It is somewhat more involved to calculate the uniform average over all priors (indexed by) α of $P(c \mid d, \alpha)$. The result is derived in Appendix D:

Theorem 8. *Assume OTS error, a vertical $P(d \mid \phi)$, homogeneous loss L, and a homogeneous test set noise process. Let α index the priors $P(\phi)$. Then the uniform average over all α of $P(c \mid d, \alpha)$ equals $\Lambda(c)/r$.*

By Corollary (2), Theorem (7) means that the average over all priors α of $P(c \mid m, \alpha)$ equals $P(c \mid m,$ uniform prior). Similarly, by Theorem (6), Theorem (8) means that the average over all priors α of $P(c \mid d, \alpha)$ equals $P(c \mid d,$ uniform prior). In this sense, whatever one's learning algorithm, one can just as readily have a prior that gives worse performance than that associated with the uniform prior as one that gives better performance.

To put this even more strongly, consider again the uniform-random learning algorithm discussed at the beginning of this section. By Theorems (7) and (8), for any learning algorithm, one can just as readily have a prior for which that algorithm performs worse than the random learning algorithm—*worse than random guessing*—as a prior for which one's algorithm performs better than the random learning algorithm.

It may be that for some particular (homogeneous) noise process, for some training sets d and target functions ϕ, $P(c \mid \phi, d)$ is not defined. This is the situation, for example, when there is no noise [d must lie on ϕ, so for any other d and ϕ, $P(c \mid \phi, d)$ is meaningless]. In such a situation, averaging over all ϕs with d fixed [as in Theorem (4)] is not well-defined. Such situations can, at the expense of extra work, be dealt with explicitly. [The result is essentially that all of the results of this section except Theorem (4) are obtained.] Alternatively, one can usually approximate the analysis for such noise processes arbitrarily well by using other, infinitesimally different noise processes, processes for which $P(c \mid \phi, d)$ is always defined.

4.3 Examples. Example 1: Say we have no noise in either training set or test set generation, and the zero-one loss $L(\cdot, \cdot)$. Fix two possible (single-valued) hypotheses, h_1 and h_2. Let learning algorithm A take in the training set d, and guess whichever of h_1 and h_2 agrees with d more often (the "majority" algorithm). Let algorithm B guess whichever of h_1 and h_2 agrees *less* often with d (the "antimajority" algorithm). If h_1 and h_2 agree equally often with d, both algorithms choose randomly between them. Then averaged over all target functions ϕ, $E(C \mid \phi, m)$ is the same for A and B.

As an example, take $n = 5$ and $r = 2$ (i.e., $\mathbf{X} = \{0, 1, 2, 3, 4\}$, and $\mathbf{Y} = \{0, 1\}$) and a uniform sampling distribution $\pi(x)$. Take m', the number of distinct elements in the training set, to equal 4. For expository purposes, I will explicitly show that the average over all ϕ of $E(C \mid \phi, m')$ is the same for A and B. [To calculate the average over all ϕ of $E(C \mid \phi, m)$, one sums the average of $E(C \mid \phi, m')\, P(m' \mid m)$ over all m'.] I will take $h_1 =$ the all 1s h, and $h_2 =$ the all 0s h.

1. There is one ϕ that is all 0s (i.e., for which for all $\mathbf{X}$ values, $\mathbf{Y} = 0$).

For that ϕ, algorithm A always picks h_2, and therefore $E(C \mid \phi, m' = 4) = 0$; algorithm A performs perfectly. For algorithm B, expected $C = 1$.

2. There are five ϕs with one 1. For each such ϕ, the probability that the training set has all four zeroes is 0.2. The value of C for such training sets is 1 for algorithm A, 0 for B. For all other training sets, $C = 0$ for algorithm A, and 1 for algorithm B. So for each of these ϕs, the expected value of C is $0.2(1) + 0.8(0) = 0.2$ for A, and $0.2(0) + 0.8(1) = 0.8$ for B.
3. There are 10 ϕs with two 1s. For each such ϕ, there is a 0.4 probability that the training set has one 1, and a 0.6 probability that it has both 1s. (It can't have no 1s.) If the training set has a single 1, so does the OTS region, and $C = 1$ for A, 0 for B. If the training set has two 1s, then our algorithms say guess randomly, so (expected) $C = 0.5$ for both algorithms. Therefore for each of these ϕs, expected $C = 0.4(1) + 0.6(.5) = 0.7$ for algorithm A, and $0.4(0) + 0.6(.5) = 0.3$ for B. Note that here B outperforms A.
4. The case of ϕs with three 1s is the same as the case of ϕs with two 1s (just with "1" replaced by "0" throughout). Similarly, four 1s = one, and five 1s = one. So it suffices to just consider the cases already investigated, where the number of 1s is zero, one, or two.
5. Adding them up, for algorithm A we have one ϕ with (expected) $C = 0$, five with $C = 0.2$, and 10 with $C = 0.7$. So averaged over all those ϕs, we get $[1(0) + 5(0.2) + 10(0.7)]/[1 + 5 + 10] = 0.5$. This is exactly the same expected error as algorithm B has: expected error for B is $[1(1) + 5(0.8) + 10(0.3)]/16 = 0.5$. QED.

See Example 5 in paper two for a related example.

Example 2: An algorithm that uses cross-validation to choose among a prefixed set of learning algorithms does no better on average than one that does not, so long as the loss function is homogeneous. In addition, cross-validation does no better than anti-cross-validation (choosing the learning algorithm with the *worst* cross-validation error) on average. In particular, the error on the validation set can be measured using a non-homogeneous loss (e.g., quadratic loss), and this result will still hold; all that is required is that we use a homogeneous loss to measure error on the test set.

Alternatively, construct the following algorithm: "If cross-validation says one of the algorithms under consideration has particularly low error in comparison to the other, use that algorithm. Otherwise, choose randomly among the algorithms." Averaged over all targets, this algorithm will do exactly as well as the algorithm that always guesses randomly among the algorithms. In this particular sense, you cannot rely on cross-validation's error estimate (unless you impose a prior over targets or some such).

Note that these results don't directly address the issue of how accurate cross-validation is as an estimator of generalization accuracy; the object of concern here is instead the error that accompanies use of cross-validation. For a recent discussion of the accuracy question (though in a non-OTS context), see Plutowski *et al.* (1994). For a more general discussion of how error and accuracy-as-an-estimator are statistically related (especially when that accuracy is expressed as a confidence interval), see Wolpert (1994a). The issue of how accurate cross-validation is as an estimator of generalization accuracy is also addressed in the discussion just below Theorem (9), and in the fixed-f results in paper two.

Example 3: Assume you are a Bayesian, and calculate the Bayes-optimal guess assuming a particular $P(f)$ [i.e., you use the $P(h \mid d)$ that would minimize the data-conditioned risk $E(C \mid d)$, if your assumed $P(f)$ were the correct $P(f)$]. You now compare your guess to that made by someone who uses a non-Bayesian method. Then the NFL theorems mean (loosely speaking) that there are as many actual priors (your assumed prior being fixed) in which the other person has a lower data-conditioned risk as there are for which your risk is lower.

Example 4: Consider any of the heuristics that people have come up with for supervised learning: avoid "over-fitting," prefer "simpler" to more "complex" models, "boost" your algorithm, "bag" it, etc. The NFL theorems say that all such heuristics fail as often (appropriately weighted) as they succeed. This is true despite formal arguments some have offered trying to prove the validity of some of these heuristics.

4.4 General Implications of the NFL Theorems. The primary importance of the NFL theorems is their implication that, for any two learning algorithms A and B, according to any of the distributions $P(c \mid d)$, $P(c \mid m)$, $P(c \mid f, d)$, or $P(c \mid f, m)$, there are just as many situations (appropriately weighted) in which algorithm A is superior to algorithm B as vice versa. So in particular, if we know that learning algorithm A is superior to B averaged over some set of targets F, then the NFL theorems tell us that B must be superior to A if one averages over all targets not in F. This is true even if algorithm B is the algorithm of purely random guessing.

Note that much of computational learning theory, much of sampling theory statistics (e.g., bias + variance results), etc., is based on quantities like $P(c \mid f, m)$, or on other quantities determined by $P(c \mid f, m)$ (see Wolpert 1994a). Similarly, conventional Bayesian analysis is concerned with $P(c \mid d)$. All of these quantities are addressed in the NFL theorems.

As a special case of the theorems, when there are only two possible values of $L(\cdot, \cdot)$, any two algorithms are even more tightly matched in behavior than Theorems (1) through (8) indicate. [An example of such an $L(\cdot, \cdot)$ is zero-one loss, for which there are only two possible values of $L(\cdot, \cdot)$, regardless of r.] Let C_1 and C_2 be the costs associated with two learning algorithms. Now $P(c_1 \mid \text{stuff}) = \sum_{c_2} P(c_1, c_2 \mid \text{stuff})$, and similarly for $P(c_2 \mid \text{stuff})$. (Examples of "stuff" are $\{d, f\}$, $\{m\}$, f-averages of these,

etc.) If $L(\cdot, \cdot)$ can take on two values, this provides us four equations (one each for the two possible values of c_1 and the two possible values of c_2) in four unknowns [$P(c_1, c_2 \mid \text{stuff})$ for the four possible values of c_1 and c_2]. Normalization provides a fifth equation. Accordingly, if we know both $P(c_1 \mid \text{stuff})$ and $P(c_2 \mid \text{stuff})$ for both possible values of c_1 and c_2, we can solve for $P(c_1, c_2 \mid \text{stuff})$ (sometimes up to some overall unspecified parameters, since our five equations are not independent). In particular, if we know that $P_{C_1}(c \mid \text{stuff}) = P_{C_2}(c \mid \text{stuff})$, then $P(c_1, c_2 \mid \text{stuff})$ must be a symmetric function of c_1 and c_2. So for all of the "stuff"s in the NFL theorems, when $L(\cdot, \cdot)$ can take on two possible values, for any two learning algorithms, $P(c_1, c_2 \mid \text{stuff})$ is a symmetric function of c_1 and c_2 (under the appropriate uniform average).[1]

All of the foregoing applies to more than just OTS error. In general IID error can be expressed as a linear combination of OTS error plus on-training set error, where the combination coefficients depend only on $d_{\mathbf{X}}$ and $\pi(x \in d_{\mathbf{X}})$. So generically, if two algorithms have the same on-training set behavior (e.g., they reproduce d exactly), the NFL theorems apply to their IID errors as well as their OTS set errors. (See also Appendix B.)

Notwithstanding the NFL theorems though, learning algorithms can differ in that (1) for particular f, or particular (nonuniform) $P(f)$, different algorithms can have different probabilities of error (this is why some algorithms tend to perform better than others in the real world); (2) for some algorithms there is a distribution-conditioning quantity (e.g., an f) for which that algorithm is optimal (i.e., for which that algorithm beats all other algorithms), but some algorithms are not optimal for any value of such a quantity; and more generally (3) for some pairs of algorithms the NFL theorems may be met by having comparatively many targets in which algorithm A is just slightly worse than algorithm B, and comparatively few targets in which algorithm A beats algorithm B by a lot. These points are returned to in paper two.

4.5 Extensions for Nonuniform Averaging. The uniform sums over f [or ϕ, or $P(\phi)$] in the NFL theorems are not necessary conditions for those theorems to hold. As an example, consider the version of the theorems for which targets are single-valued functions ϕ from $\mathbf{X}$ to $\mathbf{Y}$, perhaps with output-space noise superimposed, and where one averages over priors α. It turns out that we recover the NFL result for that scenario if we average according to any distribution over the α which is invariant under relabeling of the ϕ. We do not need to average according to the uniform distribution, and in fact can disallow all priors that are too close to the uniform prior.

More formally, we have the following variant of Theorem (7), proven in Appendix C:

[1] For more than two possible values of $L(\cdot, \cdot)$, it is not clear what happens. Nor is it clear how much of this carries over to costs C' (see Section 3) rather than C.

Corollary 3. *Assume OTS error, a vertical $P(d \mid \phi)$, homogeneous loss L, and a homogeneous test-set noise process. Let α index the priors $P(\phi)$, and let $G(\alpha)$ be a distribution over α. Assume $G(\alpha)$ is invariant under the transformation of the priors α induced by relabeling the targets ϕ. Then the average according to $G(\alpha)$ of $P(c \mid m, \alpha)$ equals $\Lambda(c)/r$.*

As a particular example of this result, define α^* to be the uniform prior, that is the vector all of whose components are equal. Then one $G(\alpha)$ that meets the assumption in Corollary (3) is the one that is constant over α except that it excludes all vectors α lying within some L^2 distance of α^* [i.e., one $G(\alpha)$ that meets the assumption is the one that excludes all priors α that are too close to being uniform]. This is because rearranging the components of a vector does not change the distance between that vector and α^*, so any $G(\alpha)$ that depends only on that distance obeys the assumption in Corollary (3).

Combined with Corollary (3), this means that $G(\alpha)$ can have structure—it can have a huge amount of structure—and we still get NFL. Alternatively, the set of allowed priors can be tiny, and restricted to priors α with a lot of structure (i.e., to priors lying far from the uniform prior), and we still get NFL. Loosely speaking, there are just as many priors that have lots of structure for which your favorite algorithm performs worse than randomly as there are for which it performs better than randomly.

An open question is whether the condition on $G(\alpha)$ in Corollary (3) is a necessary condition to have the average according to $G(\alpha)$ of $P(c \mid m, \alpha)$ equal $\Lambda(c)/r$.

Interestingly, we do not have the same kind of result when considering averages over targets f of $P(c \mid f, m)$ rather than averages over α of $P(c \mid m, \alpha)$. This is because there is no such thing as a "uniform f" that we can restrict the average away from with the same kind of implications as restricting an average away from a uniform prior. However, by Theorem (2), for any pair of algorithms, there are targets that "favor" the first of the two algorithms, and there are targets that favor of the second. So by choosing from both sets of targets, we can construct many distributions $\Gamma(f)$ that have a small support and such that the average of $P(c \mid f, m)$ according to $\Gamma(f)$ is the same for both algorithms. Indeed, an interesting open question is characterizing the set of such $\Gamma(f)$ for any particular pair of algorithms.

4.6 On Uniform Averaging. The results of the preceding subsection notwithstanding, it is natural to pay a lot of attention to the original uniform average forms of the NFL theorems. When considering those forms, it should be kept in mind that the uniform averages over f [or ϕ, or $P(\phi)$] were not chosen because there is strong reason to believe that all f are equally likely to arise in practice. Indeed, in many respects it is absurd to ascribe such a uniformity over possible targets to the real

world. Rather the uniform sums were chosen because such sums are a useful theoretical tool with which to analyze supervised learning.

For example, the implication of the NFL theorems that there is no such thing as a general-purpose learning algorithm that works optimally for all $f/P(\phi)$ is not too surprising. However, even if one already believed this implication, one might still have presumed that there are algorithms that *usually* do well and those that *usually* do poorly, and that one could perhaps choose two algorithms so that the first algorithm is usually superior to the second. The NFL theorems show that this is not the case. If all fs are weighted by their associated probability of error, then for any two algorithms A and B there are exactly as many fs for which algorithm A beats algorithm B as vice versa.

Now if one changes the weighting over fs to not be according to the algorithm's probability of error, then this result would change, and one would have a priori distinctions between algorithms. However, a priori, the change in the result could just as easily favor either A or B. Accordingly, claims that "in the real world $P(f)$ is not uniform, so the NFL results do not apply to my favorite learning algorithm" are misguided at best. Unless you can prove that the nonuniformity in $P(f)$ is well-matched to your favorite learning algorithm (rather than being "antimatched" to it), the fact that $P(f)$ may be nonuniform, by itself, provides no justification whatsoever for your use of that learning algorithm [see the inner product formula, Theorem (1), in Wolpert 1994a].

In fact, the NFL theorems for averages over priors $P(\phi)$ say (loosely speaking) that there are exactly as many priors for which any learning algorithm A beats any algorithm B as vice versa. So uniform distributions over targets are not an atypical, pathological case, out at the edge of the space. Rather they and their associated results are the average case(!). There are just as many priors for which your favorite algorithm performs worse than pure randomness as for which it performs better. [Recall the discussion just below Theorem (8).]

So for the learning scenarios considered in this section (zero-one loss, etc.) the burden is on the user of a particular learning algorithm. Unless they can somehow show that $P(\phi)$ is one of the ones for which their algorithm does better than random, rather than one of the ones for which it does worse, they cannot claim to have any formal justification for their learning algorithm.

In fact if you press them, you find that in practice very often people's assumption do not concern $P(\phi)$ at all, but rather boil down to the statement "okay, my algorithm corresponds to an assumption about the prior over targets; I make that assumption." This is unsatisfying enough a formal justification as it stands. Unfortunately though, for many algorithms, no one has even tried to write down that set of $P(\phi)$ for which their algorithm works well. This puts the purveyors of such statements in the awkward position of invoking an unknown assumption. (Moreover, for some algorithms one can show that there is *no* assumption solely

concerning targets that justifies that algorithm in all contexts. This is true of cross-validation, for example; see paper two.)

Given this breadth of the implications of the uniform-average cases, it is not surprising that uniform distributions have been used before to see what one can say a priori about a particular learning scenario. For example, the "Ugly Duckling Theorem" (Watanabe 1985) can be viewed as (implicitly) based on a uniform distribution. Another use of a uniform distribution, more closely related to the uniform distributions occurring in this paper, appears in the "problem-averaging" work of Hughes (1968). [See Waller and Jain (1978) as well for a modern view of the work of Hughes.] The words of Duda and Hart (1973) describing that work are just as appropriate here: "Of course, choosing the a priori distribution is a delicate matter. We would like to choose a 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". Let us consider some of the implications (of such an assumption)."

In this regard, note that you really would need a proof based completely on first principles to formally justify some particular (nonuniform) $P(f)$. In particular, you cannot use your "prior knowledge" (e.g., that targets tend to be smooth, that Occam's razor usually works, etc.) to set $P(f)$, without making additional assumptions about the applicability of that "knowledge" to future supervised learning problems. This is because that "prior knowledge" is ultimately an encapsulation of two things: the data set of your experiences since birth, and the data set of your genome's experiences in the several billion years it has been evolving. So if you are confronted with a situation differing at all (!) from the previous experiences of you and/or your genome, then you are in an OTS scenario. Therefore the NFL theorems apply, and you have no formal justification for presuming that your "prior knowledge" will apply off-training set (i.e., in the future).

An important example of this is the fact that even if your prior knowledge allowed you to generalize well in the past, this provides no assurances whatsoever that you can successfully apply that knowledge to some current inference problem. The fact that a learning algorithm has been used many times with great success provides no formal (!) assurances about its behavior in the future.[2] After all, assuming that how well you generalized in the past carries over to the present is formally equivalent to (a variant of) cross-validation—in both cases, one tries to extrapolate from generalization accuracy on input points for which we now know what the correct answer was, to generalization behavior in general.

Finally, it is important to emphasize that results based on averag-

[2]All of this is a formal statement of a rather profound (if somewhat philosophical) paradox: How is it that we perform inference so well in practice, given the NFL theorems and the limited scope of our prior knowledge? A discussion of some "head-to-head minimax" ideas that touch on this paradox is presented in paper two.

ing uniformly over $f/\phi/P(\phi)$ should not be viewed as normative. The uniform averaging enables us to reach conclusions that assumptions are needed to distinguish between algorithms, not that algorithms can be (profitably) distinguished without any assumptions, i.e., if such an average ends up favoring algorithm A over B (as it might for a nonhomogeneous loss function, for example), that only means one "should" use A if one has reason to believe that all f are equally likely a priori.

4.7 Other Peculiar Properties Associated with OTS Error. There are many other aspects of OTS error that, although not actually NFL theorems, can nonetheless be surprising. An example is that in certain situations the expected (over training sets) OTS error grows as the size of the training set increases, even if one uses the best possible learning algorithm, the Bayes-optimal learning algorithm [i.e., the learning algorithm which minimizes $E(C \mid d)$—see Wolpert (1994a)]. In other words, sometimes the more data you have, the less you know about the OTS behavior of ϕ, on average.

In addition, the NFL theorems have strong implications for the common use of a "test set" or "validation set" T to compare the efficacy of different learning algorithms. The conventional view is that the error measured on such a set is a sample of the full generalization error. As such, the only problem with using error on T to estimate "full error" is that error on T is subject to statistical fluctuations, fluctuations that are small if T is large enough. However if we are interested in the error for $x \notin \{d \cup T\}$, the NFL theorems tell us that (in the absence of prior assumptions) error on T is meaningless, no matter how many elements there are in T.

Moreover, as pointed out in Section (4) of the second of this pair of papers, use of test sets cannot correspond to an assumption only about targets [i.e., there is no $P(f)$ that, by itself, justifies the use of test sets]. Rather use of test sets corresponds to an assumption about both targets and the algorithms the test set is being used to choose between. Use of test sets will give incorrect results unless one has a particular relationship between the target and the learning algorithms being chosen between.

In all this, even the ubiquitous use of test sets is unjustified (unless one makes assumptions). For a discussion of this point and of intuitive arguments for why the NFL theorems hold, see Wolpert (1994a).

5 The NFL Theorems and Computational Learning Theory

This section discusses the NFL theorem's implications for and relationship with computational learning theory.

Define the empirical error

$$s \equiv \sum_{i=1}^{m} \sum_{y_H} L[y_H, d_Y(i)]\ h[d_X(i), y_H]\ \pi[d_X(i)] / \sum_{i=1}^{m} \pi[d_X(i)]$$

Sometimes the values $\pi[d_X(i)]$ in this definition are replaced by a constant; doing so has no effect on the analysis below. As an example, for zero-one loss and single-valued h, s is the average misclassification rate of h over the training set. Note that the empirical error is implicitly a function of d and h but of nothing else (π being fixed). (For deterministic learning algorithms, this reduces to being a function only of d.) So for example $P(s \mid d,f) = \int dh\, P(s \mid d,f,h)P(h \mid d) = \int dh\, P(s \mid d,h)P(h \mid d) = P(s \mid d)$.

This section first analyzes distributions over C that involve the value of s, as most of computational learning theory does. Then it analyzes OTS behavior of "membership queries" algorithms and also of "punting" algorithms (those that may refuse to make a guess), algorithms that are also analyzed in computational learning theory.

5.1 NFL Theorems Involving Empirical Error. Some of the NFL theorems carry over essentially unchanged if one conditions on s in the distribution of interest. This should not be too surprising. For example, consider the most common kind of learning algorithms, deterministic ones that produce single-valued hs. For such learning algorithms, the training set d determines the hypothesis h and therefore determines s. So specifying s in addition to d in the conditioning statement of the probability distribution provides no information not already contained in d. This simple fact establishes the NFL theorem for $P(c \mid f,d,s)$, for these kinds of learning algorithms.

More generally, first follow along with the derivation of Lemma (1), to get

Lemma 2. $P(c \mid f,d,s) = \sum_{y_H,y_F,q} \delta[c, L(y_H,y_F)]\, P(y_H \mid q,d,s)\, P(y_F \mid q,f)P(q \mid d)$

where use was made of the identities $P(y_F \mid q,f,s) = P(y_F \mid q,f)$, and $P(q \mid d,s) = P(q \mid d)$. (Both identities follow from the fact that $P_{A|B,S,D,H}[a \mid b,s(d,h),d,h)] = P(a \mid b,d,h)$ for any variables A and B.)

Continuing along with the logic that resulted in Theorem (1), we arrive at the following analogue of Theorem (1) (that holds even for non-deterministic learning algorithms, capable of guessing non-single-valued hs):

For homogeneous loss L, the uniform average over all f of $P(c \mid f,d,s)$ equals $\Lambda(c)/r$.

Unfortunately, one cannot continue paralleling the analysis in Section (3) past this point, to evaluate quantities like the uniform average over all f of $P(c \mid f,s,m)$. The problem is that whereas $P(d \mid f,m)$ is independent of $f(x \notin d_X)$ (for a vertical likelihood), the same need not be true of $P(d \mid f,s,m)$. Indeed, often there are fs for which $P(c_{OTS} \mid f,s,m)$ is not defined; for no d sampled from that f will an h be produced that has

error s with that d. In such scenarios the uniform f-average of $P(c_{\text{OTS}} \mid f, s, m)$ is not defined. Moreover, the set of f for which $P(c_{\text{OTS}} \mid f, s, m)$ is defined may vary with s. The repercussions of this carry through for any attempt to create s-conditioned analogs of the NFL theorems. (A counter-intuitive example of how the NFL theorems need not hold for s-conditioned distributions is presented in Appendix C.)

In fact, it is hard to say anything general about $P(c \mid f, s, m)$. In particular, it is not always the (peculiar) case that higher s results in lower c_{OTS} if f is fixed, as in the example in Appendix C. To see this, consider the scenario given there with a simple change in the learning algorithm. For the new learning algorithm, if all input elements of the training set, d_X, are in some region Ξ, then an hypothesis h is produced that happens to equal the target f, whereas for any other d_Xs, there are errors both on and off d_X. So if $s = 0$, we know that $c_{\text{OTS}} = 0$. But if $s > 0$, we know that $c_{\text{OTS}} > 0$; raising s from 0 has raised expected C_{OTS}.

Now consider $P(c \mid s, d)$ for uniform $P(f)$, where it is implicitly assumed that for at least one h for which $P(h \mid d) \neq 0$, the empirical error is s, so $P(s, d) \neq 0$. For this quantity we do have an NFL result that holds for any learning algorithm (see Appendix C):

Theorem 9. *For homogeneous L, OTS error, a vertical likelihood, and uniform $P(f)$, $P(c \mid s, d) = \Lambda(c)/r$.*

The immediate corollary is that for homogeneous L, OTS error, a vertical likelihood, and uniform $P(f)$, $P(c \mid s, m) = \Lambda(c)/r$, independent of the learning algorithm.

It is interesting to note that a uniform $P(f)$ can give NFL for $P(c \mid s, m)$ even though a uniform average over f of $P(c \mid f, s, m)$ does not. This illustrates that one should exercise care in equating the basis of NFL for f-conditioned distributions [Theorem (2)] with having a uniform prior.

An immediate question is how Theorem (9) can hold despite the example above where as s shrinks $E(C_{\text{OTS}} \mid f, s, m)$ grows, for any target f. The answer is that $P(c \mid s, m) = \int df\, P(c \mid f, s, m)\, P(f \mid s, m)$. Even if for any *fixed* target f the quantity $P(c \mid f, s, m)$ gets biased toward lower cost c as the empirical error s is raised, this does not mean that the integral exhibits the same behavior.

As an aside, it should be noted that the only property of s needed by Theorem (9) or its corollary is that $P(s \mid d, f) = P(s \mid d)$. In addition to holding for the random variable S, this property will hold for any random variable σ that is a function only of d for the algorithms under consideration. So in particular, we can consider using cross-validation to choose among a set of one or more deterministic algorithms. Define σ as the cross-validation errors of the algorithms involved. Since for a fixed set of deterministic algorithms σ is a function solely of d, we see that for a uniform $P(f)$, σ is statistically independent from C; there is no information contained in the set of cross-validation errors that has bearing on generalization error. In this sense, unless one makes an explicit

assumption for $P(f)$, cross-validation error has no use as an estimate of generalization error.

5.2 Compatibility with Vapnik–Chervonenkis Results. The fact that $P(c \mid s.m) = \Lambda(c)/r$ (under the appropriate conditions) means that $P(c \mid s.m) = P(c \mid m)$ under those conditions [see Corollary (1)]. This implies that $P(s \mid c.m)$ is independent of cost c. So C_{OTS} and empirical error S are statistically independent, for uniform $P(f)$, homogeneous L, and a vertical likelihood. Indeed, in Appendix B in Wolpert (1992) there is an analysis of the case where we have a uniform sampling distribution $\pi(\cdot)$, zero-one loss, binary $\mathbf{Y}$, and $n/m \to \infty$ (so $C''_{\text{OTS}} \to C'_{\text{IID}}$; see Appendix B of this paper). It is there proven that $E(C'_{\text{IID}} \mid s.m) = 1/2$, independent of s.

In accord with this, one expects that C'_{OTS} and S are independent for uniform $P(f)$. On the other hand, Vapnik–Chervonenkis (uniform convergence) theory tells us that $P(c'_{\text{IID}} - s \mid m)$ is biased toward small values of $c'_{\text{IID}} - s$ for low-VC dimension generalizers, and large m. This is true for any prior $P(f)$, and therefore in particular for a uniform prior. It is also true even when $n \gg m$, so that C'_{OTS} and C'_{IID} closely approximate each other.

It should be emphasized that there is no contradiction between these VC results and the NFL theorems. Independence of s and c'_{OTS} does not imply that s and c'_{OTS} can differ significantly. For example, both the VC results and the NFL theorems would hold if for many f $P(c'_{\text{OTS}} \mid f.m)$ and $P(s \mid f.m)$ were independent but were both tightly clumped around the same value, ζ.

Now let us say we have an instance of such a "clumping" phenomenon, but do not know ζ (ζ being determined by (the unknown) f, among other things). We might be tempted to take the observed value of s as an indicator of the likely value of ζ. In turn, we might wish to view this likely value of ζ as an indicator of the likely value of c'_{OTS}. In this way, having observed a particular value of s, we could infer something about c'_{OTS} (e.g., that it is unlikely to differ from that observed value of s). However Theorem (9) says that this reasoning is illegal [at least for uniform $P(f)$]. Statistical independence is statistical independence; knowing the value of s tells you nothing whatsoever about the value of c'_{OTS} (see Wolpert 1994a for further discussion of how independence of s and c'_{OTS} is compatible with the VC theorems).

Intuitively, many of the computational learning theory results relating empirical error s and generalization error c'_{IID} are driven by the fact that s is formed by sampling c'_{IID} (see Wolpert 1994a). However, for OTS c' the empirical error s cannot be viewed as a sample of c'. Rather s and c'_{OTS} are on an equal footing. Indeed, for single-valued targets and hypotheses, and no noise, s and c'_{OTS} are both simply the value c'_{IID} has when restricted to a particular region in $\mathbf{X}$. (The region is $d_{\mathbf{X}}$ for s, $\mathbf{X} - d_{\mathbf{X}}$ for c'_{OTS}.) In this sense, there is symmetry between s and c'_{OTS} (symmetry absent for s

and c'_{IID}). Given this, it should not be surprising that for uniform $P(f)$, the value of s tells us nothing about the value of c'_{OTS} and vice versa.

5.3 Implications for Vapnik–Chervonenkis Results. The s-independence of the results presented above has strong implications for the uniform convergence formalism for investigating supervised learning (Vapnik 1982; Vapnik and Bottou 1993; Anthony and Biggs 1992; Natarajan 1991; Wolpert 1994a). Consider zero-one loss, where the empirical error s is very low and the training set size m is very large. Assume that our learning algorithm has a very low VC dimension. Since s is low and m large, we might hope that that low VC dimension confers some assurance that our generalization error will be low, independent of assumptions concerning the target. (This is one common way people try to interpret the VC theorems.)

However according to the results presented above, low s, large m, and low VC dimension, by themselves, provide no such assurances concerning OTS error (unless one can somehow a priori rule out a uniform $P(f)$—not to mention rule out any other prior having even more dire implications for generalization performance). This is emphasized by the example given above where a tight confidence interval on the probability of c'_{OTS} differing from s arises solely from $P(c'_{\text{OTS}} \mid m)$ and $P(s \mid m)$ being peaked about the same value; s and c'_{OTS} are statistically independent, so knowing s tells you nothing concerning c'_{OTS}. Indeed, presuming c'_{OTS} is small due only to the fact that s, m, and the learning algorithm's VC dimension are small can have disastrous real-world consequences (see the example concerning "We-Learn-It Inc." in Wolpert 1994a).

Of course, there are many other conditioning events one could consider besides the ones considered in this paper. And, in particular, there are many such events that involve empirical errors. For example, one might investigate the behavior of the uniform f-average of $P(c \mid s_A, s_B, m, f)$, where s_A and s_B are the empirical errors for the two algorithms A and B considered in Example (1) in Section (3).

It may well be that for some of these alternative conditioning events involving empirical errors, one can find a priori distinctions between learning algorithms, dependences on s values, or the like. Although such results would certainly be interesting, one should be careful not to ascribe too much practical significance to them. In the real world, it is almost always the case that we know d and h in full, not simply functions of them like the empirical error. In such a scenario, it is hard to see why one would be concerned with a distribution of the form $P[c \mid \mathit{function}(d), h]$, as opposed to distributions of the form $P(c \mid d)$ [or perhaps $P(c \mid d, h)$, or the f-average of $P(c \mid d, f)$, or some such]. So since the NFL theorems say there is no a priori distinction between algorithms as far as $P(c \mid d)$ is concerned, it is hard to see why one should choose between algorithms based on distributions of the form $P[c \mid \mathit{function}(d), h]$, if one does indeed know d in full.

5.4 Implications of the NFL Theorems for Active Learning Algorithms. Active learning (aka "query-based learning," or "membership queries") is where the learner decides what the points d_X will be. Usually this is done dynamically; as one gets more and more training examples, one uses those examples to determine the "optimal" next choices of $d_X(i)$.

As far as the EBF is concerned, the only difference between active learning and traditional supervised learning is in the likelihood. Rather than IID likelihoods like that in equation (3.1), in active learning each successive $d_X(i)$ is a function of the $(i-1)$ pairs $\{d_X(j = 1.i-1).d_Y(j = 1.i-1)\}$, with the precise functional dependence determined by the precise active learning algorithm being used.

So long as it is true that $P[d_Y(m) \mid d_X(m).f]$ is independent of $f[x \neq d_X(m)]$, active learning has a vertical likelihood (see Appendix C). So all of the negative implications of the NFL theorems apply just as well to active learning as IID likelihood learning, and in particular apply to the kinds of active learning discussed in the computational learning community.

5.5 Implications of the NFL Theorems for "Punting" Learning Algorithms. Some have advocated using algorithms that have an extra option besides making a guess. This option is to "punt," i.e., refuse to make a guess. As an example, an algorithm might choose to punt because it has low confidence in its guess (say for VC theory reasons). It might appear that, properly constructed, such algorithms could avoid making bad guesses. If this were the case, it would be an assumption-free way of ensuring that *when one guesses*, the guesses are good. (One would have traded in the ability to always make a guess to ensure that the guesses one does make are good ones.) In particular, some have advocated using algorithms that add elements to d adaptively until (and if) they can make what they consider to be a safe guess.

However the simple fact that a particular punting algorithm has a small probability of making a poor guess, by itself, is no reason to use that algorithm. After all, the completely useless algorithm that always punts has zero probability of making a poor guess. Rather what is of interest is how well the algorithm performs when it does guess, and/or how accurate its punt-signal warning is as an indicator that to make a guess would result in large error. To analyze this, I will slightly modify the definition of punting algorithms so that they always guess, but also always output a punt / no punt signal (and perhaps ask for more training set elements), based deterministically only on the d at hand. The issue at hand then is how the punt / no punt signal is statistically correlated with C.

Examine *any* training set d for which some particular algorithm outputs a no punt signal. By the NFL theorems, for such a d, for uniform $P(f)$, a vertical $P(d \mid f)$, and a homogeneous OTS error, $P(c \mid d)$ is the same as that of a random generalizer, i.e., under those conditions, $P(c \mid d.\text{no punt}) = \Lambda(c)/r$. As a direct corollary, $P(c \mid m.\text{no punt}) = \Lambda(c)/r$. It

follows that $P(c \mid \text{no punt}) = \Lambda(c)/r$ (assuming the no punt signal arises while OTS error is still meaningful, so $m' < n$).

Using the same kind of reasoning though, we also get $P(c \mid \text{punt}) = \Lambda(c)/r$, etc. So there is no statistical correlation between the value of the punt signal and OTS error. Unless we assume a nonuniform $P(f)$, even if our algorithm "grows" d until there is a no punt signal, the value of the punt / no punt signal tells us nothing about C. Similar conclusions follow from comparing a punting algorithm to its "scrambled" version, as in the analysis of nonhomogeneous error (see paper two).

In addition, let A and B be two punting algorithms that are identical in when they decide to output a punt signal, but B guesses randomly for all test inputs $q \notin d_X$. Then for the usual reasons, A's distribution over OTS error is, on average, the same as that of B, i.e., no better than random. This is true even if we condition on having a no punt signal.

One nice characteristic of some punting algorithms—the characteristic exploited by those who advocate such algorithms—is that there can be some prior-free assurances associated with them. As an example, for all targets f, the probability of such an algorithm guessing and making an error in doing so is very small [see classes (1) and (2) below]: $\forall f$, for sufficiently large m and nonnegligible ε, $P(c_{\text{OTS}} > \varepsilon, \text{no punt} \mid f, m)$ is tiny.

However $P(c_{\text{OTS}} > \varepsilon, \text{no punt} \mid f, m)$ in fact equals 0 for the always-punt algorithm. So one might want to also consider other distributions like $P(c_{\text{OTS}} > \varepsilon \mid \text{no punt}, f, m)$ or $P(c_{\text{OTS}} < 1 - \varepsilon, \text{no punt} \mid f, m)$ to get a more definitive assessment of the algorithm's utility. Unfortunately though, both of these distributions are highly f-dependent. (This illustrates that the f-independent aspects of the punting algorithm mentioned in the previous paragraph do not give a full picture of the algorithm's utility.)

In addition, other f-independent results hardly inspire confidence in the idea of making a guess only when there is a no punt signal. As an illustration, restrict things so that both hypotheses h and targets are single-valued (and therefore targets are written as functions ϕ), and there is no noise. $\mathbf{Y}$ is binary, and we have zero-one loss. Let the learning algorithm always guess the all 0s function, h^*. The punt signal is given if d_Y contains at least one non-zero value. Then for the likelihood of (3.1), uniform $\pi(x)$, and $n \gg m$, we have the following result, proven in Appendix E:

Theorem 10. *For the h^* learning algorithm, for all targets ϕ such that $\phi(x) = 0$ for more than m distinct x, $E(C_{\text{OTS}} \mid \phi, \text{punt}, m) \leq E(C_{\text{OTS}} \mid \phi, \text{no punt}, m)$.*

For $n \gg m$, essentially all ϕ meet the requirement given in Theorem (10); for such n and m, we do better to follow the algorithm's guessing advice when we are told not to than we are told the guessing is good!

In many respects, the proper way to analyze punting algorithms is given by decision theory. First, assign a cost to punting. (Formally, this

just amounts to modifying the form of $P(c \mid f, h, d)$ for the case where h and d lead to a punt signal.) This cost should not be less than the minimal no-punting cost, or the optimal algorithm is to never guess. Similarly, it should not be more than the maximal no-punting cost, or the optimal algorithm never punts. Given such a punting cost, the analysis of a particular punting algorithm consists of finding those $P(f)$ such that $E(C_{\text{OTS}} \mid m)$ is "good" (however defined). In lieu of such an analysis, one can find those $P(f)$ such that $E(C_{\text{OTS}} \mid \text{no punt}, m) < E(C_{\text{OTS}} \mid \text{punt}, m)$ (e.g., one can analyze whether priors that are uniform in some sphere centered on h^* and zero outside of it result in this inequality). Such analyses—apparently never carried out by proponents of punting algorithms—are beyond the scope of this paper however. (In addition, they vary from punting algorithm to punting algorithm.)

5.6 Intuitive Arguments Concerning the NFL Theorems and Punting Algorithms. Consider again the algorithm addressed in Theorem (10). For this algorithm, there are two separate kinds of ϕ:

1. ϕ such that the algorithm will almost always punt for a d of sufficient size sampled from ϕ, or
2. ϕ such that the algorithm has tiny expected error when it chooses not to punt.

(Targets ϕ with almost no xs such that $\phi(x) = 1$ are in the second class, and other targets are in the first class.)

It might seem that this breakdown justifies use of the algorithm. After all, for large enough m', if the target is such that there is a nonnegligible probability that the algorithm does not punt, it is not in class 1, so if it does not punt error will be tiny. Thus it would seem that *whatever the target* (or prior over targets), if the algorithm has not punted, we can be confident in its guess. [Similar arguments can be made when the two classes distinguish sets of $P(\phi)$s rather than ϕs.]

However, if we restate this, the claim is that $E(C \mid \text{no punt}, m)$ is tiny for sufficiently large m, for any prior over targets $P(\phi)$. (Note that for $n \gg m$ and non-pathological $\pi(\cdot)$, m' is unlikely to be much less than m.) This would imply, in particular, that it is tiny for uniform $P(\phi)$. However from the preceding subsection we know that this is not true. So we appear to have a paradox.

To resolve this paradox, consider using our algorithm and observing the no punt signal. Now restate (1) and (2) carefully: In general, either

1. The target is such that for sufficiently large m' the algorithm will almost always punt, *but when it does not punt, it usually makes significant errors*, or
2. The target is such that the algorithm has tiny expected error when it chooses not to punt.

Now there are many more ϕs in class 1 than in class 2. So even though the probability of our no-punt signal is small for each of the ϕs in class 1 individually, when you multiply by the number of such ϕ, you see that the probability of being in class 1, given that you have a no-punt signal, is not worse than the probability of being in class 2, given the same signal. In this sense, the signal gains you nothing in determining in which class you are in, and therefore in determining likely error.[3]

So at a minimum, one must assume that $P(\phi)$ is not uniform to have justification for believing the punt/no punt signal. Now one could argue that a uniform $P(\phi)$ is highly unlikely when there is a no-punt signal, i.e., $P[\text{no punt} \mid \alpha = \text{uniform}\, P(\phi), m]$ is very small, and that this allows one to dismiss this value of α if we see a no punt signal. Formally though, α is a hyperparameter, and should be marginalized out: it is axiomatically true that $P(\phi) = \int d\alpha\, P(\phi \mid \alpha) P(\alpha)$ and is fixed beforehand, independent of the data. So the presence/absence of a punt signal cannot be used to "infer" something about $P(\phi)$, formally speaking [see the discussions of hierarchical Bayesian analysis and empirical Bayes in Berger (1985) and Bernardo and Smith (1994)]. More generally, the NFL theorems allow us to "jump a level," so that classes 1 and 2 refer to αs rather than ϕs. And at this new level, we again run into the fact that there are many more elements in class 1 than in class 2.

To take another perspective, although the likelihood $P(\text{no punt} \mid \text{class}, m)$ strongly favors class 2, the posterior need not. Lack of appreciation for this distinction is an example of how computational learning theory relies almost exclusively on likelihood-driven calculations, ignoring posterior calculations.

It may be useful to directly contrast the intuition behind the class 1–2 reasoning and that behind the NFL theorems: The class 1–2 logic says that given a ϕ with a nonnegligible percentages of 1s, it's hugely unlikely to get all 0s in a large random data set. Hence, so this intuitive reasoning goes, if you get all 0s, you can conclude that ϕ does not have a nonnegligible percentages of 1s, and therefore you are safe in guessing 0s outside the training set. The contrasting intuition: say you are given some particular training set, say of the first K points in $\mathbf{X}$, together with associated $\mathbf{Y}$ values. Say the $\mathbf{Y}$ values happen to be all 0s. Obviously, without some assumption concerning the coupling of ϕs behavior over the first K points in $\mathbf{X}$ with its behavior outside of those points, ϕ could have any conceivable behavior outside of those points. So the fact that it is all 0s has no significance, and cannot help you in guessing.

[3] All that is being argued in this discussion of classes (1) and (2) is that the absence of a punt signal does not provide a reason to believe error is good. This argument does not directly address whether the presence of a punt signal gives you reason to believe you are in class (1), and therefore is correlated with bad error. The explanation of why there is no such correlation is more subtle than simply counting the number of ϕs in each class. It involves the fact that there are actually a continuum of classes, and that for fixed ϕ, raising s (so as to get a punt signal) lowers OTS (!) error.

It should be emphasized that none of the reasoning of this subsection directly addresses the issue of whether the punting algorithm has good "head-to-head minimax" OTS behavior in some sense (see paper two). That is an issue that has yet to be thoroughly investigated. In addition, recall that no claims are being made in this paper about what is (not) reasonable in practice; punting algorithms might very well work well in the real world. Rather the issue is what can be formally established about how well they work in the real world without making any assumptions concerning targets.

5.7 Differences between the NFL Theorems and Computational Learning Theory. Despite the foregoing, there are some similarities between the NFL theorems and computational learning theory. In particular, when all targets are allowed—as in the NFL theorems—PAC bounds on the error associated with $s = 0$ are extremely poor (Blumer *et al.* 1987, 1989; Dietterich 1990; Wolpert 1994a). However there are important differences between the NFL theorems and this weak-PAC-bounds phenomenon.

1. For the most part, PAC is designed to give positive results. In particular, this is the case with the PAC bounds mentioned above. (More formally, the bounds in question give an upper bound on the probability that error exceeds some value, not a lower bound.) However lack of a positive result is not the same as a negative result, and the NFL theorems are full-blown negative results.
2. PAC (and indeed all of computational learning theory) has nothing to say about these data (i.e., Bayesian) scenarios. They only concern data-averaged quantities. PAC also is primarily concerned with polynomial versus exponential convergence issues, i.e., asymptotics of various sorts. The NFL theorems hold even if one does not go to the limit, and hold even for these data scenarios. [See also Wolpert (1994a) for a discussion of how PAC's being exclusively concerned with convergence issues renders its real-world meaningfulness debatable, at best.]
3. The PAC bounds in question can be viewed as saying there is no universally good learning algorithm. They say nothing about the possibility of whether some algorithm 1 may be better than some other algorithm 2 in most scenarios. As a particular example, nothing in the PAC literature suggests that there are as many (appropriately weighted) fs for which a boosted learning algorithm (Drucker *et al.* 1993; Shapire 1990) performs *worse* than its unboosted version as there are for which the reverse is true.
4. The PAC bounds in question do not emphasize the importance of a vertical likelihood; they do not emphasize the importance of homogeneous noise when the target is a single-valued function; they do

not emphasize the importance of whether the loss function is homogeneous; they do not invoke "scrambling" (see paper two) for nonhomogeneous loss functions (indeed, they rarely consider such loss functions); they do not concern averaging over pairs of hs (in the sense of Section (4) of paper two), etc. In all this, they are too general. Note that this overgenerality extends beyond the obvious problem that they are "(sampling) distribution free." Rather they are too general in that they are independent of many of the features of a supervised learning problem that are crucially important.

5. Computational learning theory does not address OTS error. Especially when m is not infinitesimal in comparison to n and/or $\pi(x)$ is highly nonuniform, computational learning theory results are changed significantly if one uses OTS error (see Wolpert 1994a). And even for infinitesimal m and fairly uniform $\pi(x)$, many distributions behave very differently for OTS rather than IID error (see Section 5.2).

Appendix A. Detailed Exposition of the EBF

This Appendix discusses the EBF in some detail. Since it is the goal of this paper to present as broadly applicable results as possible, care is taken in this Appendix to discuss how a number of different learning scenarios can be cast in terms of the EBF.

Notation

- In general, unless indicated otherwise, random variables are written using upper case letters. A particular instantiation value of such a random variable is indicated using the corresponding lower case letter. Note though that some quantities (e.g., parameters like the size of the spaces) are neither random variables nor instantiations of random variables, so their written case carries no significance.
- When clarity is needed, the argument of a $P(\cdot)$ will not be used to indicate what the distribution is; rather a subscript will denote the distribution. For example, $P_F(h)$ means the prior over the random variable F (targets), evaluated at the value h (a particular hypothesis). This is common statistics notation. (Note that with conditioning bars, this notation leads to expressions like "$P_{A|B}(c \mid d)$," meaning the probability of random variable A conditioned on variable B, evaluated at values c and d, respectively.)
- Also in accord with common statistics notation, "$E(A \mid b)$" will be used to mean the expectation value of A given $B = b$, i.e., to mean $\int da\, a\, P(a \mid b)$. (Sums replace integrals if appropriate.) This means in particular that anything not specified is averaged over. So for example, $E(A \mid b) = \int dc\, da\, a\, P(a \mid b, c)\, P(c \mid b) = \int dc\, E(a \mid b, c)\, P(c \mid$

b). When it is obvious that their value is assumed fixed and what it is fixed to, sometimes I will not specify variables in conditioning arguments.

- I will use n and r to indicate the (countable though perhaps infinite) number of elements in the set **X** (the input space) and the set **Y** (the output space), respectively. (**X** and **Y** are the only case in this paper where capital letters do not indicate random variables.) Such cases of countable **X** and **Y** are the simplest to present, and always obtain in the real world where data are measured with finite precision instruments and are manipulated on finite size digital computers.

A generic **X** value is indicated by x, and a generic **Y** value by y. Sometimes I will implicitly take **Y** and/or **X** to be sets of real numbers, sometimes finely spaced. (This is the case when talking about the "expected value" of a **Y**-valued random variable, for example.)

The Primary Random Variables

- In this paper, the "true" or "target" relationship between (test set) inputs and (test set) outputs is taken to be an **X**-conditioned distribution over **Y** [i.e., intuitively speaking, a $P(y \mid x)$]. In other words, where S_r is defined as the r-dimensional unit simplex, the "target distribution" is a random variable mapping $\mathbf{X} \rightarrow S_r$. Since **X** and **Y** are simply sets and not themselves random variables, this is formalized as follows:

Let F be a random variable taking values in the n-fold cartesian product space of simplices S_r. Let f be a particular instantiation of that variable, i.e., an element in the n-fold cartesian product space of simplices S_r. Then f can be viewed as a Euclidean vector, with indices given by a value $x \in \mathbf{X}$ and $y \in \mathbf{Y}$. Accordingly, we can indicate a component of f by writing $f(x, y)$. So for all x, y, $f(x, y) \geq 0$, and for any fixed x, $\sum_y f(x, y) = 1$.

This defines the random variable F. The formal sense in which this F can be viewed as an "**X**-conditioned distribution over **Y**" arises in how it is statistically related to certain other random variables (specified below) taking values in **X** and in **Y**.

- In a similar fashion, the generalizer's hypothesis is an "**X**-conditioned distribution over **Y**," i.e., the hypothesis random variable H takes values in the n-fold cartesian product space of simplices S, and components of any instantiation h of H can be indicated by $h(x, y)$.
- If for all x, $h(x, y)$ is a Kronecker delta function (over y), h is called "single-valued," and similarly for f. In such a case, the distribution in question reduces to a single-valued function from **X** to **Y**.
- The value d of the training set random variable is an ordered set of m input-output pairs, or "examples." Those pairs are indicated by

$d_{\mathbf{X}}(i)$, $d_{\mathbf{Y}}(i)\{i = 1 \ldots m\}$. The set of all input values in d is $d_{\mathbf{X}}$ and similarly for $d_{\mathbf{Y}}$. m' is the number of distinct values in $d_{\mathbf{X}}$.

- The cost C is a real-valued random variable.
- The primary random variables are such target distributions F, such hypothesis distributions H, training sets D, and real-world "cost" or "error" values C measuring how well one's learning algorithm performs. They are "coupled" to supervised learning by imposing certain conditions on the relationship between them, conditions that are discussed next.

The Relationship between C, F, and H, Mediated by Q, Y_F, and Y_H. It will be useful to relate C to F and H using three other random variables. "Testing" (involved in determining the value of C) is done at the **X** value given by the **X**-valued random variable Q. **Y** values associated with the hypothesis and Q are given by the **Y**-valued random variable Y_H (with instantiations y_H), and **Y** values associated with the target and Q are given by the **Y**-valued random variable Y_F (with instantiations y_F).

All of this is formalized as follows.

- The F random variable parameterizes the Q-conditioned distribution over Y_F : $P(y_F \mid f, q) = f(q, y_F)$. In other words, f determines how test set elements y_F are generated for a test set point q. So Y_F and Q are the random variables whose relationship to F allows F to be intuitively viewed as an "**X**-conditioned distribution over **Y**"—see above.
- The variable Y_H meets similar requirements: $P(y_H \mid h, q) = h(q, y_H)$, and this relationship between Y_H, Q, and H is what allows one to view H as intuitively equivalent to an "**X**-conditioned distribution over **Y**."
- For the purposes of this paper, the random variable cost C is defined by $C = L(Y_H, Y_F)$, where $L(\cdot, \cdot)$ is called a "loss function." As examples, zero-one loss has $L(a, b) = 1 - \delta(a, b)$, where $\delta(a, b)$ is the Kronecker delta function, and quadratic loss has $L(a, b) = (a - b)^2$. (Zero-one loss is assumed in almost all of computational learning theory.)

  It is important to note though that in general C need not correspond to such a loss function. For example, "logarithmic scoring" has $c = -\sum_y f(q, y) \ln[h(q, y)]$, and does not correspond to any $L(y_F, y_H)$.
- For many Ls the sum over y_F of $\delta[c, L(y_H, y_F)]$ is some function $\Lambda(c)$, independent of y_H. I will call such Ls "homogeneous." Intuitively, such Ls have no a priori preference for one **Y** value over another. As examples, the zero-one loss is homogeneous. So is the squared difference between y_F and y_H if they are viewed as angles, $L(y_F, y_H) = [(y_F - y_H) \bmod \pi]^2$.

Note that one can talk of an L's being homogeneous for certain values of c. For example, the quadratic loss is not homogeneous over all c, but it is for $c = 0$. The results presented in this paper that rely on homogeneity of L usually hold for a particular c so long as L is homogeneous for that c, even if L is not homogeneous for all c.

The Relationship between F, D and Q

- Note that f is a distribution governing test set data (it governs the outputs associated with q), and in general it need not be the same as the distribution governing training set data. Unless explicitly stated otherwise though, I will assume that both training sets and test sets are generated via f.
- Often when training and testing sets are generated by the same $P(y \mid x)$, the training set is formed by iterating the following "independent identically distributed" (IID) procedure: Choose $\mathbf{X}$ values according to a "sampling distribution" $\pi(x)$, and then sample f at those points to get associated $\mathbf{Y}$ values.[4] More formally, this very common scheme is equivalent to the following "likelihood," presented previously as equation 3.1:

$$\begin{aligned} P(d \mid f) &= P(d_{\mathbf{Y}} \mid f, d_{\mathbf{X}})\, P(d_{\mathbf{X}} \mid f) \\ &= P(d_{\mathbf{Y}} \mid f, d_{\mathbf{X}})\, P(d_{\mathbf{X}}) \quad \text{(by assumption)} \\ &= \prod_{i=1}^{m} \{\pi[d_{\mathbf{X}}(i)] f[d_{\mathbf{X}}(i), d_{\mathbf{Y}}(i)]\} \end{aligned}$$

There is no a priori reason for $P(d \mid f)$ to have this form, however. For example, in "active learning" or "query-based" learning, successive values (as i increases) of $d_{\mathbf{X}}(i)$ are determined by the preceding values of $d_{\mathbf{X}}(i)$ and $d_{\mathbf{Y}}(i)$. As another example, typically $P(d \mid f)$ will not obey equation (3.1) if testing and training are not governed by the same $P(y \mid x)$. (Recall that f governs the generation of test sets.) To see this, let t be the random variable $P(y \mid x)$ governing the generation of training sets. Then $P(d \mid f) = \int dt\, P(d \mid t)\, P(t \mid f)$. Even if $P(d \mid t) = \prod_{i=1}^{m} \{\pi[d_{\mathbf{X}}(i)] t[d_{\mathbf{X}}(i), d_{\mathbf{Y}}(i)]\}$, unless $P(t \mid f)$ is a delta function about $t = f$, $P(d \mid f)$ need have the form specified in equation 3.1.

- I will say that $P(d \mid f)$ is "vertical" if it is independent of the values of $f(x \notin d_{\mathbf{X}})$. Any likelihood of the form given in equation (3.1) is vertical, by inspection. In addition, as discussed in Section 5, active

[4] In general, π itself could be a random variable that can be estimated from the data, that is perhaps coupled to other random variables (e.g., f), etc. However here I make the usual assumption in the neural net and computational learning literature that π is fixed. This is technically known as a "filter likelihood," and has powerful implications (see Wolpert 1994b).

learning usually has a vertical likelihood. However, some scenarios in which $t \neq f$ do not have vertical likelihoods.[5]

- In the case of "IID error" (the conventional error measure), $P(q \mid d) = \pi(q)$. In the case of OTS error, $P(q \mid d) = [\delta(q \notin d_X)\pi(q)]/[\sum_q \delta(q \notin d_X)\pi(q)]$, where $\delta(z) \equiv 1$ if z is true, 0 otherwise. Strictly speaking, OTS error is not defined when $m' = n$.

  Where appropriate, subscripts OTS or IID on c will indicate which kind of $P(q \mid d)$ is being used.

Function + Noise Targets

- In this paper I will consider in some detail those cases where we only allow those f that can be viewed as some single-valued function ϕ taking $\mathbf{X}$ to $\mathbf{Y}$ with a fixed noise process in $\mathbf{Y}$ superimposed.[6] To do this, I will (perhaps only implicitly) fix a noise function N that is a probability distribution over $\mathbf{Y}$, conditioned on $\mathbf{X} \times \mathbf{Y}$; N is a probability distribution over y_F, conditioned on the values of q and $\phi(q)$. [Note that there are r^n such functions $\phi(\cdot)$.]

Given $N(\cdot)$, each ϕ specifies a unique f_ϕ, via $P(y_F \mid f_\phi, q) = f_\phi(q, y_F) = N[y_F \mid q, \phi(q)] = P(y_F \mid q, \phi)$. Accordingly, all the usual rules concerning f apply as well to ϕ. [For example, $P(h \mid d, \phi) = P(h \mid d)$.] When I wish to make clear what ϕ sets f, I will write f_ϕ, as above; ϕ simply serves as an index on f. [In general, depending on $N(\cdot)$, it might be that more than one ϕ labels the same f, but this will not be important for the current analysis.] So when I say something like "vertical $P(d \mid \phi)$" it is implicitly understood that I mean vertical $P(d \mid f_\phi)$.

- When I say that I am "only allowing" these kinds of f, I will mean that whenever "f" is written, it is assumed to be related to a ϕ in this manner—all other f implicitly have an infinitesimal prior probability.
- Note that the $N(\cdot)$ introduced here is the noise process operating in the generation of the test set, and need not be the same as the noise

[5] As an example, assume that f is some single-valued function from $\mathbf{X}$ to $\mathbf{Y}$, ϕ, so that $P(y_F \mid f_\phi, q) = \delta[y_F, \phi(q)]$. However, assume that d is created by corrupting ϕ with both noise in $\mathbf{X}$ and noise in $\mathbf{Y}$. This can be viewed as a "function + noise" scenario where the noise present in generating the training set is absent in testing. (This case is discussed in some detail below.)

As an example of such a scenario, viewing any particular pair of $\mathbf{X}$ and $\mathbf{Y}$ values from the training set as random variables X_t and Y_t, one might have $Y_t = \sum_{X'} \gamma(X_t, X')\phi(X') + \varepsilon$, where X' is a dummy X variable, $\gamma(\cdot, \cdot)$ is a convolutional process giving noise in $\mathbf{X}$, and ε is a noise process in $\mathbf{Y}$. (Strictly speaking, this particular kind of $\mathbf{Y}$-noise requires that $r = \infty$, as otherwise $\sum_{x'} \gamma(x, x')\phi(x') + \varepsilon$ might not lie in $\mathbf{Y}$.)

For this scenario, $t \neq f$. In addition, $P(d \mid f)$ does not have the form given in equation 3.1. In particular, due to the convolution term, $P(d \mid f)$ will depend on the values of $f = \phi$ for $x \notin d_X$; the likelihood for this scenario is not vertical.

[6] Noise in $\mathbf{X}$ of the form mentioned in footnote 5 will not be considered in this paper. The extension to analyze such noise processes is fairly straightforward however.

process in the generation of the training set. As an example, it is common in the neural net literature to generate the training set by adding noise to a single-valued function from **X** to **Y**, $\phi(\cdot)$, but to measure error by how well the resulting h matches that underlying $\phi(\cdot)$, not by how well Y_H values sampled from h match Y_F values formed by adding noise to $\phi(\cdot)$. In the ϕ-N terminology, this would mean that although $P(d \mid f)$ may be formed by corrupting some function $\phi(\cdot)$ with noise (in either **X** and/or **Y**), $P(y_F \mid f, q)$, which is used to measure test set error, is determined by a noise-free $N(\cdot)$, $N[y_F \mid q, \phi(q)] = \delta[y_F, \phi(q)]$.

- Of special importance will be those noise-processes for which for each q, the uniform ϕ-average of $P(y_F \mid q, \phi)$ is independent of y_F. (Note this does not exclude q-dependent noise processes). I will call such a (test-set) noise process "homogeneous." Intuitively, such noise processes have no a priori preference for one **Y** value over another. As examples, the noise-free testing mentioned just above is homogeneous, as is a noise process that when it takes in a value of $\phi(q)$, produces the same value with probability z and all other values with (identical) probabilities $(1 - z)/(r - 1)$.

Coupling All This to Supervised Learning

- Any (!) learning algorithm (or "generalizer") is simply a distribution $P(h \mid d)$. It is "deterministic" if the same d always gives the same h [i.e., if for fixed d, $P(h \mid d)$ is a delta function about one particular h].
- There are many equalities that are assumed in supervised learning, but that do not merit explicit delineation. For example, it is implicitly assumed that $P(h \mid q, d) = P(h \mid d)$, and therefore that $P(q \mid d, h) = P(q \mid d)$.
- One assumption that does merit explicit delineation is that $P(h \mid f, d) = P(h \mid d)$ (i.e., the learning algorithm only sees d in making its guess, not f). This means that $P(h, f \mid d) = P(h \mid d)\, P(f \mid d)$, and therefore $P(f \mid h, d) = P(f \mid d)$.

As an example of the importance of this assumption, note that it implies that $P(y_F \mid y_H, d, q) = P(y_F \mid d, q)$.

Proof. Expand $P(y_F \mid y_H, d, q) = \int df\, f(q, y_F) P(f \mid d, q) \int dh\, h(q, y_H) P(h \mid f, d, q)$. Since $P(h \mid f, d, q) = P(h \mid d)$, this integral is proportional to $\int df\, f(q, y_F)\, P(f \mid d, q)$, where the proportionality constant depends on d, y_H, and q. However $\int df\, f(q, y_F)\, P(f \mid d, q) = P(y_F \mid d, q)$. Due to normalization, this means that the proportionality constant equals 1, and we have established the proposition. QED.

Our assumption does not imply that $P(y_F \mid y_H, d) = P(y_F \mid d)$, however. Intuitively, for a fixed learning algorithm, knowing y_H and d tells you

something about q, and therefore (in conjunction with knowledge of d) something about y_F, that d alone does not.

- The "posterior" is the Bayesian inverse of the likelihood, $P(f \mid d)$. The phrase "the prior" usually refers to $P(f)$.
- Some schemes can be cast into this framework in more than one way. As an example, consider softmax (Bridle 1989), where each output neuron indicates a different possible event, and the real values the neurons take in response to an input are interpreted as input-conditioned probabilities of the associated events. For this scheme one could either (1) take $\mathbf{Y}$ to be the set of "possible events," so that the h produced by the algorithm is not single-valued, or (2) take $\mathbf{Y}$ to be (the computer's discretization of) the real-valued vectors that the set of output neurons can take on, in which case h is single-valued, and $\mathbf{Y}$ itself is interpreted as a space of probability distributions. Ultimately, which interpretation one adopts is determined by the relationship between C and H. (Such relationships are discussed below.)

"Generalization Error"

- Note that $E(C \mid f, h, d) = \sum_{y_H, y_F, q} E(C \mid f, h, d, y_H, y_F, q)\, P(y_H, y_F, q \mid f, h, d)$. Due to our definition of C, the first term in the sum equals $L(y_H, y_F)$. The second term equals $P(y_H \mid h, q, f, d, y_F)\, P(y_F \mid q, f, d, h)\, P(q \mid d, f, h)$. This in turn equals $h(q, y_H) f(q, y_F)\, P(q \mid f, h, d)$. In addition, $P(q \mid f, h, d) = P(q \mid d)$ always in this paper. Therefore $E(C \mid f, h, d) = \sum_{y_H, y_F, q} L(y_H, y_F)\, h(q, y_H) f(q, y_F)\, P(q \mid d)$.

In much of supervised learning, an expression like that on the right-hand side of this equation is called the "generalization error." In other words, instead of the error C used here, in much of supervised learning one uses an alternative error C', defined by $C'(f, h, d) \equiv E(C \mid f, h, d)$, i.e., $P(c' \mid f, h, d) = \delta[c', E(C \mid f, h, d)]$.

- Note that in general, the set of allowed values of C is not the same as the set of allowed values of C'. In addition, distributions over C do not set those over C'. For example, knowing $P(c \mid d)$ need not give $P(c' \mid d)$ or vice versa.[7] However many branches of supervised

[7] If there are only two possible $L(\cdot, \cdot)$ values (for example), $P(c' \mid d)$ does give $P(c \mid d)$. This is because $P(c' \mid d)$ gives $E(C' \mid d) = E(C \mid d)$ (see below in Appendix A), and since there are two possible costs, $E(C \mid d)$ gives $P(c \mid d)$. It is for more than two possible cost values that the distributions $P(c' \mid d)$ and $P(c \mid d)$ do not determine one another. In fact, even if there are only two possible values of $L(\cdot, \cdot)$, so that $P(c' \mid d)$ sets $P(c \mid d)$, it does not follow that $P(c \mid d)$ sets $P(c' \mid d)$. As an example, consider this case where $n = m' + 2$, and we have zero-one loss. Assume that given some d, $P(f \mid d)$ and $P(h \mid d)$ are such that either h agrees exactly with f for OTS q or the two never agree, with equal probability. This means that for zero-one OTS error, $P(c \mid d) = \delta(c, 0)/2 + \delta(c, 1)/2$. However we would get the same distribution if all four possible agreement relationships between h and f for the off-training set q were possible, with equal probabilities. And in that

learning theory (e.g., much of computational learning theory) are concerned with quantities of the form "$P(\text{error} > \varepsilon \mid \ldots)$."[8] For such quantities, whether one takes "error" to mean C or (as is conventional) C' may change the results, and in general one cannot directly deduce the result for C from that for C' (or vice versa).

Where appropriate, subscripts OTS or IID on c' will indicate which kind of $P(q \mid d)$ is being used.

- Fortunately, most of the results derived in this paper apply equally well to both probabilities of C and probabilities of C'. For reasons of space though, I will work out the results explicitly only for C. However, note that we can immediately equate expectations of C that are not conditioned on q, y_H, or y_F with the same expectations of C'. For example,

$$\begin{aligned} E(C \mid d) &= \int dhdf\, E(C \mid f, h, d)\, P(f, h \mid d) \\ &= \int dhdf\, C'(f, h, d)\, P(f, h \mid d) \\ &= \int dhdf\, E(C' \mid f, h, d)\, P(f, h \mid d) = E(C' \mid d) \end{aligned}$$

So when cast in terms of expectation values, any (appropriately conditioned) results automatically apply to C' as well as C.

Miscellaneous

- For most purposes, it is implicitly assumed that no probabilities equal zero *exactly* (although some probabilities might be infinitesimal). That way we have never have to worry about dividing by probabilities, and in particular never have to worry about whether conditional probabilities are well-defined. So as an example, phrases like "noise-free" are taken to mean infinitesimal noise rather than exactly zero noise. Similarly, where needed, integrals over f are implicitly restricted away from fs having one or more components equal to zero.

- It is important to note that in general, for nonpathological $\pi(\cdot)$, in the limit where $n \gg r$, distributions over c'_{IID} are identical to distributions over c'_{OTS}. In this sense theorems concerning OTS error immediately carry over to IID error. This is proven formally in Appendix B.

---

second case, we would have the possibility of C' values that are impossible in the first case (e.g., $c = 1/2$). QED.

[8] In general, whether "error" means C or C', this quantity is of interest only if the number of values error can have is large. So for example, it is of interest for C' if r is large and we have quadratic loss.

Appendix B. Proof That Distributions Over C'_{IID} Equal Those Over C'_{OTS} Whenever $n \gg r$, for Nonpathological $\pi(\cdot)$

To prove the assertion, with slight abuse of terminology write $C'_{\text{IID}} = C'_{\text{OTS}}\pi(\mathbf{X} - d_{\mathbf{X}}) + C'_{\text{TS}}\pi(d_{\mathbf{X}})$, where "TS" means error on the training set, defined in the obvious way, and $\pi(A) \equiv \sum_{x \in A} \pi(x)$ (see Wolpert *et al.* 1995). Then for any set of one or more random variables Z taking value z, we have

$$\begin{aligned}
&P(c'_{\text{IID}} \mid z) \\
&= \sum_d \int dc'_{\text{OTS}}\, dc'_{\text{TS}}\, P(c'_{\text{IID}} \mid c'_{\text{OTS}}, c'_{\text{TS}}, d, z) \\
&\quad \times P(c'_{\text{OTS}}, c'_{\text{TS}}, d \mid z) \\
&= \sum_d \int d_{c'_{\text{OTS}}}\, dc'_{\text{TS}}\, \delta[c'_{\text{IID}} - c'_{\text{OTS}}\, \pi(\mathbf{X} - d_{\mathbf{X}}) - c'_{\text{TS}}\, \pi(d_{\mathbf{X}})] \\
&\quad \times P(c'_{\text{OTS}}, c'_{\text{TS}}, d \mid z) \\
&= \sum_d \int d_{c'_{\text{OTS}}}\, dc'_{\text{TS}}\, \delta[c'_{\text{IID}} - c'_{\text{OTS}}\, \pi(\mathbf{X} - d_{\mathbf{X}}) - c'_{\text{TS}}\, \pi(d_{\mathbf{X}})] \\
&\quad \times \int df\, dh\, P(c'_{\text{OTS}}, c'_{\text{TS}} \mid d, f, h, z, m)\, P(d, f, h \mid z)
\end{aligned}$$

Now again abuse terminology slightly and write

$$\begin{aligned}
P(c'_{\text{OTS}}, c'_{\text{TS}} \mid d, f, h, z) \;=\; & \delta[c'_{\text{OTS}} - C'_{\text{OTS}}(d, f, h)] \\
& \times \delta[c'_{\text{TS}} - C'_{\text{TS}}(d, f, h)]
\end{aligned}$$

where the statistical dependencies of C'_{OTS} and C'_{TS} are made explicit by writing them as functions. Plugging in we get

$$\begin{aligned}
&P(c'_{\text{IID}} \mid z) \\
&= \sum_d \int df\, dh\, \delta[c'_{\text{IID}} - C'_{\text{OTS}}(d, f, h)\, \pi(\mathbf{X} - d_{\mathbf{X}}) - C'_{\text{TS}}(d, f, h)\, \pi(d_{\mathbf{X}})] \\
&\quad \times P(d, f, h \mid z)
\end{aligned}$$

Define $\varepsilon \equiv \max_{d_{\mathbf{X}}} \pi(d_{\mathbf{X}})$, so $\min_{d_{\mathbf{X}}} \pi(\mathbf{X} - d_{\mathbf{X}}) = 1 - \varepsilon$. Now whenever $n \gg m$, so long as there are no sharp peaks in $\pi(\cdot)$, $\varepsilon \to 0$. However, because a delta function is not a continuous function, taking the limit as $\varepsilon \to 0$ of our expression for $P(c'_{\text{IID}} \mid z)$ is not immediately equivalent to setting the $\pi(\mathbf{X} - d_{\mathbf{X}})$ and $\pi(d_{\mathbf{X}})$ inside the delta function to 1 and to 0, respectively. We can circumvent this difficulty rather easily though. To do that, first define

$$\delta \equiv \max_{\{d_{\mathbf{X}}, f, h\}} \{C'_{\text{OTS}}(d, f, h)[1 - \pi(\mathbf{X} - d_{\mathbf{X}})] - C'_{\text{TS}}(d, f, h)\pi(d_{\mathbf{X}})\}$$

Then write

$$P_{C'_{\text{IID}}|Z}(c'_{\text{IID}} + \kappa \mid z) = \sum_d \int df\, dh\, \delta[c'_{\text{IID}} - C'_{\text{OTS}}(d,f,h)] \times P(d,f,h \mid z)$$

for some κ where $|\kappa| \leq \delta$.

Now for nonpathological z, $P(c'_{\text{IID}} \mid z)$ is a continuous function of c'_{IID} as n and/or $\pi(\cdot)$ are varied. (Recall, in particular, that in this paper, no event has exactly zero probability; see Appendix A.) So for such a z, for ε sufficiently small, $\delta \to 0$ and therefore $\kappa \to 0$, and we can approximate

$$P(c'_{\text{IID}} \mid z) = \sum_d \int df\, dh\, \delta[c'_{\text{IID}} - C'_{\text{OTS}}(d,f,h)] \times P(d,f,h \mid z)$$

But this just equals $P_{C'_{\text{OTS}}|z}(c'_{\text{IID}} \mid z)$. So for $n \gg m$ and nonpathological z and $\pi(\cdot)$, the distribution over c'_{IDD} is the same as that over c'_{OTS}. QED.

Appendix C. Miscellaneous Proofs

For clarity of the exposition, several of the more straightforward proofs in the paper are collected in this appendix.

Proof of Lemma 1. Write $P(c \mid f,d) = \sum_{y_H,y_F,q} P(c \mid y_H,y_F,q,f,d)\, P(y_H \mid y_F,q,f,d)\, P(y_F,q \mid f,d)$. Rewriting the summand, we get $P(c \mid f,d) = \sum_{y_H,y_F,q} \delta[c, L(y_H,y_F)]\, P(y_H \mid y_F,f,q,d)\, P(y_F,q \mid f,d)$.

Now $P(y_H \mid y_F,f,q,d) = \int dh\, P(y_H \mid y_F,h,f,q,d)\, P(h \mid y_F,f,q,d) = \int dh\, P(y_H \mid h,q,d)\, P(h \mid q,d)$ [see point (11) in the EBF section]. This just equals $P(y_H \mid q,d)$. [However it is *not* true in general that $P(y_H \mid y_F,d) = P(y_H \mid d)$ (see Wolpert *et al.* 1995).] Plugging in gives the result. QED.

Proof of the Claim Concerning the "Random Learning Algorithm," Made Just Below Lemma (1). By Lemma 1, $P(c \mid f,d) = \sum_{y_H,y_F,q} \delta[c, L(y_H,y_F)]\, P(y_H \mid q,d)\, P(y_F \mid q,f)\, P(q \mid d)$. However, for OTS error $q \notin d_{\mathbf{X}}$, and therefore for the random learning algorithm, for all q and d in our sum, $P(y_H \mid q,d) = 1/r$, independent of y_H (recall that there are r elements in $\mathbf{Y}$). If we have a symmetric homogeneous loss function, this means that we can replace $\sum_{y_H} \delta[c, L(y_H,y_F)]\, P(y_H \mid q,d)$ with $\Lambda(c)/r$. Since this is independent of d and f, $P(c \mid d) = \Lambda(c)/r$ for all training sets d, as claimed. QED.

Proof of the "Implication of Lemma (1)," Made Just Below Lemma (1). Uniformly average the expression for $P(c \mid f,d)$ in Lemma (1) over all targets f. The only place f occurs in the sum in Lemma (1) is in the third term, $P(y_F \mid q,f)$. Therefore our average replaces that third term with some function $\text{func}(y_F,q)$. By symmetry though, the uniform f-average of that third term in the sum must be the same for all test set inputs

q and outputs y_F. Accordingly $\text{func}(y_F, q)$ is some constant. Now the sum over y_F of this constant must equal 1 [to evaluate that sum of the f-average of $P(y_F \mid q, f)$, interchange the sum over y_F with the average over f]. Therefore our constant must equal $1/r$. The implication claimed is now immediate. QED.

Proof of Theorem (2). We can replace the sum over all q that gives $P(c \mid f, d)$ [Lemma (1)] with a sum over those q lying outside of d_X. Accordingly, for such a $P(q \mid d)$, $P(c \mid f, d)$ is independent of the values of $f(x \in d_X)$. (For clarity, the second argument of f is being temporarily suppressed.)

Noting that $P(d \mid f)$ is vertical, next average both sides of our equation for $P(c \mid f, m)$ uniformly over all f and pull the f-average inside the sum over d. Since $P(c \mid f, d)$ and $P(d \mid f)$ depend on separate parts of f [namely $f(x \notin d_X)$ and $f(x \in d_X)$, respectively], we can break the average over f into two successive averages, one operating on each part of f, and thereby get

$$\sum_d \left[\int df(x \notin d_X)\, P(c \mid f, d) \int df(x \in d_X)\, P(d \mid f) \right] / \int df(x \notin d_X)\, df(x \in d_X)\, 1$$

But since $P(c \mid f, d)$ is independent of the values of $f(x \in d_X)$, uniformly averaging it over all $f(x \notin d_X)$ is equivalent to uniformly averaging it over all f. By Theorem (1), such an average is independent of d. Therefore we can pull that average out of the sum over d, and get Theorem (2). QED.

Proof of Theorem (3). Write $P(c \mid d)$ for a uniform $P(f) \propto \int df\, P(c \mid d, f)\, P(d \mid f)$, where the proportionality constant depends on d. Break up the integral over f into an integral over $f(x \mid \in d_X)$ and one over $f(x \notin d_X)$, exactly as in the proof of Theorem (2). Absorb $\int df(x \in d_X)\, P(d \mid f)$ into the overall (d-dependent) proportionality constant. By normalization, the resultant value of our constant must be the reciprocal of $\int df(x \notin d_X) 1$. QED.

Proof of Theorem (7). $\int d\alpha\, P(c \mid m, \alpha) = \int d\alpha [\sum_\phi P(\phi \mid m, \alpha) P(c \mid m, \alpha, \phi)]$, where the integral is restricted to the r^n-dimensional simplex. This can be rewritten as $\int d\alpha [\sum_\phi \alpha_\phi P(c \mid \phi, m, \alpha)]$, since we assume that the probability of ϕ has nothing to do with the number of elements in d. Similarly, once ϕ is fixed, the probability that $C = c$ does not depend on α, so our average equals $\int d\alpha [\sum_\phi \alpha_\phi P(c \mid \phi, m)]$. Write this as $\sum_\phi P(c \mid \phi, m) [\int d\alpha\, \alpha_\phi]$. By symmetry, the term inside the square brackets is independent of ϕ. Therefore the average over all $P(\phi)$ of $P(c \mid m)$ is proportional to $\sum_\phi P(c \mid \phi, m)$. Using Theorem (5) and normalization, this establishes Theorem (7). QED.

Proof of Corollary (3). Follow along with the proof of Theorem (7). Instead of $\int d\alpha\, \alpha_\phi$, we have $\int d\alpha\, G(\alpha)\, \alpha_\phi$. (For present purposes, the delta and Heaviside functions that force α to stay on the unit simplex

are implicit.) By assumption, $G(\alpha)$ is unchanged under the bijection of replacing all vectors α_i with new vectors identical to the old, except that the components for $i = o$ and $i = o'$ are interchanged. This is true for all o' and o. Accordingly, our integral is independent of o, which suffices to prove the result. QED.

Proof of Theorem (9). To evaluate $P(c \mid s.d)$ for uniform $P(f)$, write it as $\int df P(c \mid s.d.f)P(f \mid d.s)$. Next write $P(f \mid d.s) = P(s \mid f.d)P(f \mid d)/P(s \mid d)$. Note though that $P(s \mid f.d) = P(s \mid d)$ (see beginning of Section 5), and recall that we are implicitly assuming that $P(s \mid d) \neq 0$). So we get $P(c \mid s.d) = \int df P(c \mid s.d.f)P(d \mid f)$, up to an overall d-dependent proportionality constant. Now proceed as in the proof of Theorem (2) by breaking the integral into two integrals, one over $f(x \in d_{\mathbf{X}})$, and one over $f(x \notin d_{\mathbf{X}})$. The result is $P(c \mid s.d) = \Lambda(c)/r$, up to an overall d-dependent proportionality constant. By normalization, that constant must equal 1. This establishes Theorem (9). QED.

Example of Non-NFL Behavior of s-Conditioned Distributions. Let $P(h \mid d) = \delta(h.h^*)$ for some h^*, let $\pi(x)$ be uniform, use zero-one loss, assume a noise-free IID likelihood, and take $m = 1$. Then we can write $E(C_{\text{OTS}} \mid s.f.m = 1) = E(C_{\text{OTS}} \mid s.f.m = 1.h = h^*) = [nC'_{\text{IID}}(f.h^*) - s]/(n-1)$. [Note that C'_{IID} is independent of d, and that for zero-one loss $n \times C'_{\text{IID}}(f.h^*)$ is the number of disagreements between h^* and f over all of $\mathbf{X}$.] No matter what f is, this grows as s shrinks. Since C_{OTS} can have only two values, this means that as s grows, $P(c_{\text{OTS}} \mid f.s.m = 1)$ gets biased toward the lower of the two possible values of c_{OTS}. So we do not have NFL behavior for the uniform average over f of $P(c \mid f.s.m)$—that average depends on the empirical error s.

Proof That Active Learning Has a Vertical Likelihood. Let d_k refer to the first k input–output pairs in the training set d, and $d(i)$ to the ith such pair. Then $P(d_m \mid f) = P[d(m) \mid f.d_{m-1}]P(d_{m-1} \mid f) = P[d_{\mathbf{Y}}(m) \mid d_{\mathbf{X}}(m).f.d_{m-1}]P[d_{\mathbf{X}}(m) \mid f.d_{m-1}]P(d_{m-1} \mid f)$. By hypothesis, in active learning $P[d_{\mathbf{X}}(m) \mid f.d_{m-1}] = P[d_{\mathbf{X}}(m) \mid d_{m-1}]$. So long as it is also true that $P[d_{\mathbf{Y}}(m) \mid d_{\mathbf{X}}(m).f.d_{m-1}] = P[d_{\mathbf{Y}}(m) \mid d_{\mathbf{X}}(m).f]$ is independent of $f[x \neq d_{\mathbf{X}}(m)]$, by induction we have a vertical likelihood.

Appendix D. Proof of Theorem (8)

The task before us is to calculate the average over all α of $P(c \mid d.\alpha)$. To that end, write the average as (proportional to) $\int d\alpha[\sum_\phi P(\phi \mid d.\alpha)P(c \mid o.d.\alpha)]$, where as usual the integral is restricted to the r^n-dimensional simplex. Rewrite this integral as $\int d\alpha[\sum_\phi P(o \mid \alpha)P(c \mid \phi.d.\alpha)P(d \mid \phi.\alpha)/P(d \mid \alpha)] = \int d\alpha[\sum_\phi \alpha_\phi P(c \mid o.d)P(d \mid \phi)]/[\sum_{\phi'} \alpha_{\phi'} P(d \mid \phi')]$, where ϕ' is a

dummy ϕ value. Rewrite this in turn as

$$\sum_{\phi} P(c \mid \phi, d) P(d \mid \phi) \left\{ \int d\alpha \frac{\alpha_\phi}{\sum_{\phi'} \alpha_{\phi'} P(d \mid \phi')} \right\}$$

As in the proof of Theorem (2), break up ϕ into two components, ϕ_1 and ϕ_2, where ϕ_1 fixes the values of ϕ over the $\mathbf{X}$ values lying inside $d_{\mathbf{X}}$, and ϕ_2 fixes it over the values outside of $d_{\mathbf{X}}$. We must find how the terms in our sum depend on ϕ_1 and ϕ_2.

First, write $P(c \mid \phi, d) = \sum_h P(h \mid d) P(c \mid h, \phi, d)$. By definition, for OTS error $P(c \mid h, \phi, d)$ is independent of ϕ_1. This allows us to write $P(c \mid \phi, d) = P(c \mid \phi_2, d)$.

Next, since we are restricting attention to vertical likelihoods, $P(d \mid \phi)$ depends only on ϕ_1. So we can write the term in the curly brackets as $\int d\alpha [\alpha_{\phi_1 \phi_2} / \sum_{\phi_1' \phi_2'} \alpha_{\phi_1' \phi_2'} P(d \mid \phi_1')] = \int d\alpha [\alpha_{\phi_1 \phi_2} / \sum_{\phi_1'} \alpha_{\phi_1'} P(d \mid \phi_1')]$ with obvious notation. Since we are assuming that for no ϕ does $P(d \mid \phi)$ equal zero exactly, the denominator sum is always nonzero.

Now change variables in the integral over α by rearranging the ϕ_2 indices of α. In other words, ϕ_1 and ϕ_2 are a pair of discrete-valued vectors, and α is a real-valued vector indexed by a value for ϕ_1 and one for ϕ_2; transform α so that its dependence on ϕ_2 is rearranged in some arbitrary—though invertible—fashion. Performing this transformation is equivalent to mapping the space of all ϕ_2 vectors into itself in a one-to-one manner. The Jacobian of this transformation is 1, and the transformation does not change the functional form of the constraint forcing α to lie on a simplex (i.e., $\sum_{\phi_1 \phi_2''} \alpha_{\phi_1 \phi_2''} = 1$ and for all $\phi_1 \phi_2''$, $\alpha_{\phi_1 \phi_2''} \geq 0$, where double-prime indicates the new ϕ_2 indices). So expressed in this new coordinate system, the integral is $\int d\alpha \{\alpha_{\phi_1, \phi_2^*} / \sum_{\phi_1'} \alpha_{\phi_1'} P(d \mid \phi_1')\}$, where ϕ_2^* is a new index corresponding to the old index ϕ_2. Since this integral must have the same value as our original integral, and since ϕ_2^* is arbitrary, we see that that integral is independent of ϕ_2, and therefore can only depend on the values of d and ϕ_1.

This means that we can rewrite our sum over all ϕ as

$$\sum_{\phi_1 \phi_2} P(c \mid \phi_2, d) P(d \mid \phi_1) \, \text{func}_1\{\phi_1, d\}$$

for some function "$\text{func}_1(\cdot)$." In other words, the α-average of $P(c \mid d, \alpha)$ is proportional to $\sum_{\phi_2} P(c \mid \phi_2, d)$, where the proportionality constant depends on d. Since $P(c \mid \phi, d) = P(c \mid \phi_2, d)$ (see above), our sum is proportional to $\sum_{\phi_1 \phi_2} P(c \mid \phi, d) = \sum_{\phi} P(c \mid \phi, d)$. By Theorem (4), this sum equals $\Lambda(c)/r$.

So the uniform α-average of $P(c \mid d, \alpha) = \text{func}_2(d) \, \Lambda(c)/r$ for some function "$\text{func}_2(\cdot)$." Since $\sum_c P(c \mid d, \alpha) = 1$, the sum over C values of the uniform α-average of $P(c \mid d, \alpha)$ must be independent of d (it must equal 1). Therefore $\text{func}_2(d)$ is independent of d. Since we know that $\Lambda(c)/r$ is properly normalized over c, we see that $\text{func}_2(d)$ in fact equals 1. QED.

Appendix E. Proof of Theorem (10)

First use the fact that given ϕ, d_X determines whether there is a punt signal, to write

$$E[C_{OTS} \mid \phi, (\text{no}) \text{ punt}, m] = \sum_{d_X} E(C_{OTS} \mid \phi, d_X) \times P[d_X \mid (\text{no}) \text{ punt}, \phi, m] \quad \text{(E.1)}$$

Next, without loss of generality, let the xs for which $\phi(x) = 0$ be $1, \ldots, k$, so that $\phi(x) = 1$ for $x = k+1, \ldots, n$. Then $P(d_X \mid \text{no punt}, \phi, m) = 0$ unless all the $d_X(i) \leq k$. Since $\pi(x)$ is uniform, and d is ordered and perhaps has repeats, the value of $P(d_X \mid \text{no punt}, \phi, m)$ when all the $d_X(i) \leq k$ is k^{-m}. Similarly, $P(d_X \mid \text{punt}, \phi, m) = 0$ unless at least one of the $d_X(i) > k$, and when it is nonzero it equals some constant set by k and m.

It's also true that $E(C_{OTS} \mid \phi, d_X)$ is not drastically different if one considers d_Xs with a different m'. Accordingly, our summand does not vary drastically between d_Xs of one m' and d_Xs of another. Since $n \gg m$ and $\pi(x)$ is uniform though, almost all of the terms in the sum have $m' = m$. Pulling this all together, we see that to an arbitrarily good approximation (for large enough n relative to m), we can take $m' = m$. So E.1 becomes

$$E[C_{OTS} \mid \phi, (\text{no}) \text{ punt}, m] = \sum_{d_X} E(C_{OTS} \mid \phi, d_X) \times P[d_X \mid (\text{no}) \text{ punt}, m, m' = m] \quad \text{(E.2)}$$

Now consider conditioning on "no punt," in which case all the $d_X(i) \leq k$. For such a situation, for $m' = m$, $E(C_{OTS} \mid \phi, d_X) = (n - k)/(n - m)$. In contrast, consider having a punt signal, in which case at least one $d_X(i) > k$. Now $E(C_{OTS} \mid \phi, d_X) \leq (n - k - 1)/(n - m) < (n - k)/(n - m)$.

Combining this with E.2, we get $E(C_{OTS} \mid \phi, \text{punt}, m) < E(C_{OTS} \mid \phi, \text{no punt}, m)$. QED.

Acknowledgments

I would like to thank Cullen Schaffer, Wray Buntine, Manny Knill, Tal Grossman, Bob Holte, Tom Dietterich, Karl Pfleger, Mark Plutowski, Bill Macready, Bruce Mills, David Stork, and Jeff Jackson for interesting discussions. This work was supported in part by the Santa Fe Institute and by TXN Inc.

References

Anthony M., and Biggs N. 1992. *Computational Learning Theory*. Cambridge University Press, Cambridge.

Berger, J. 1985. *Statistical Decision Theory and Bayesian Analysis.* Springer-Verlag, Berlin.

Berger, J., and Jeffreys, W. 1992. Ockham's razor and Bayesian analysis. *Am. Sci.* **80**, 64–72.

Bernardo, J. Smith, A. 1994. *Bayesian Theory.* John Wiley, New York.

Blumer, A., et al. 1987. Occam's razor. *Inform. Process. Lett.* **24**, 377–380.

Blumer, A., et al. 1989. Learnability and Vapnik-Chervonenkis dimension. *J. ACM* **36**, 929–965.

Bridle, J. 1989. Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In *Neuro-Computing: Algorithms, Architectures, and Applications,* F. Fougelman-Soulie and J. Herault eds., Springer-Verlag, Berlin.

Dietterich, T. 1990. Machine learning. *Annu. Rev. Comput. Sci.* **4**, 255–306.

Drucker, H. et al. 1993. Improving performance in neural networks using a boosting algorithm. In *Neural Information Processing Systems 5,* S. Hanson et al. eds. Morgan Kaufmann, San Mateo, CA.

Duda, R., and Hart, P. 1973. *Pattern Classification and Scene Analysis.* John Wiley, New York.

Hughes, G. 1968. On the mean accuracy of statistical pattern recognizers. *IEEE Transac. Inform. Theory* **IT-14**, 55–63.

Kearns, M. J. 1992. Towards efficient agnostic learning. In *Proceedings of the 5th Annual Workshop on Computational Learning Theory,* ACM Press, New York.

Mitchell, T. 1982. Generalization as search. *Artif. Intell.* **18**, 203–226.

Mitchell T., and Blum, A. 1994. *Course Notes for Machine Learning,* CMU.

Murphy, P., and Pazzani, M. 1994. Exploring the decision forest: An empirical investigation of Occam's razor in decision tree induction. *J. Artif. Intell. Res.* **1**, 257–275.

Natarajan, B. 1991. *Machine Learning: A Theoretical Approach.* Morgan Kaufmann, San Mateo, CA.

Perrone, M. 1993. Improving regression estimation: Averaging methods for variance reduction with extensions to general convex measure optimization. Ph.D. thesis, Brown Univ., Physics Dept.

Plutowski, M. 1994. Cross-validation estimates integrated mean squared error. In *Advances in Neural Information Processing Systems 6,* Cowan et al. eds. Morgan Kaufmann, San Mateo, CA.

Schaffer, C. 1993. Overfitting avoidance as bias. *Machine Learn.* **10**, 153–178.

Schaffer, C. 1994. A conservation law for generalization performance. In *Machine Learning: Proceedings of the Eleventh International Conference,* Cohen and Hirsh, eds. Morgan Kaufmann, San Mateo, CA.

Schapire, R. 1990. The strength of weak learnability. *Machine Learn.* **5**, 197–227.

Vapnik, V. 1982. *Estimation of Dependences Based on Empirical Data.* Springer-Verlag, Berlin.

Vapnik, V., and Bottou, L. 1993. Local algorithms for pattern recognition and dependencies estimation. *Neural Comp.* **5**, 893–909.

Waller, W., and Jain, A. 1978. On the monotonicity of the performance of Bayesian classifiers. *IEEE Transact. Inform. Theory* **IT-24**, 392–394.

Watanabe, S. 1985. *Pattern Recognition: Human and Mechanical.* John Wiley, New York.

Weiss, S. M., and Kulikowski, C. A. 1991. *Computer Systems that Learn.* Morgan Kaufmann, San Mateo, CA.

Wolpert, D. 1992. On the connection between in-sample testing and generalization error. *Complex Syst.* **6**, 47–94.

Wolpert, D. 1993. *On Overfitting Avoidance as Bias.* Tech. Rep. SFI TR 93-03-016.

Wolpert, D. 1994a. The relationship between PAC, the Statistical Physics framework, the Bayesian framework, and the VC framework. In *The Mathematics of Generalization,* D. Wolpert ed., Addison-Wesley, Reading, MA.

Wolpert, D. 1994b. Filter likelihoods and exhaustive learning. In *Computational Learning Theory and Natural Learning Systems: Volume II,* S. Hanson et al. eds. MIT Press, Cambridge, MA.

Wolpert, D. 1995. On the Bayesian "Occam factors" argument for Occam's razor. In *Computational Learning Theory and Natural Learning Systems: Volume III,* T. Petsche et al. eds. MIT Press, Cambridge, MA.

Wolpert, D., and Macready, W. 1995. *No Free Lunch Theorems for Search.* Tech. Rep. SFI TR 95-02-010. Submitted.

Wolpert, M., Grossman, T., and Knill, E. 1995. Off-training-set error for the Gibbs and the Bayes optimal generalizers. Submitted.

Received August 18, 1995; accepted February 14, 1996.

This article has been cited by:

1. Chrysovalantis Gaganis, Fotios Pasiouras, Michael Doumpos, Constantin Zopounidis. 2010. Modelling banking sector stability with multicriteria approaches. *Optimization Letters* **4**:4, 543-558. [CrossRef]
2. Robert L. Goldstone, David Landy. 2010. Domain-Creating Constraints. *Cognitive Science* **34**:7, 1357-1377. [CrossRef]
3. A. V. Kelarev, J. L. Yearwood, P. Watters, X. Wu, J. H. Abawajy, L. Pan. 2010. Internet security applications of the Munn rings. *Semigroup Forum* **81**:1, 162-171. [CrossRef]
4. A. V. KELAREV, P. WATTERS, J. L. YEARWOOD. 2009. REES MATRIX CONSTRUCTIONS FOR CLUSTERING OF DATA. *Journal of the Australian Mathematical Society* **87**:03, 377. [CrossRef]
5. Edwin Lughofer, James E. Smith, Muhammad Atif Tahir, Praminda Caleb-Solly, Christian Eitzinger, Davy Sannen, Marnix Nuttin. 2009. Human–Machine Interaction Issues in Quality Control Based on Online Image Classification. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans* **39**:5, 960-971. [CrossRef]
6. Albert Orriols-Puig, Ester Bernadó-Mansilla. 2009. Evolutionary rule-based systems for imbalanced data sets. *Soft Computing* **13**:3, 213-225. [CrossRef]
7. D. M. Rocke, T. Ideker, O. Troyanskaya, J. Quackenbush, J. Dopazo. 2009. Papers on normalization, variable selection, classification or clustering of microarray data. *Bioinformatics* **25**:6, 701-702. [CrossRef]
8. Joanna J. Bryson. 2008. Embodiment versus memetics. *Mind & Society* **7**:1, 77-94. [CrossRef]
9. Michael Doumpos, Constantin Zopounidis. 2007. Model combination for credit risk assessment: A stacked generalization approach. *Annals of Operations Research* **151**:1, 289-306. [CrossRef]
10. Ralph van Dinther, Roy D. Patterson. 2006. Perception of acoustic scale and size in musical instrument sounds. *The Journal of the Acoustical Society of America* **120**:4, 2158. [CrossRef]
11. D.H. Wolpert, W.G. Macready. 2005. Coevolutionary Free Lunches. *IEEE Transactions on Evolutionary Computation* **9**:6, 721-735. [CrossRef]
12. David R. R. Smith, Roy D. Patterson, Richard Turner, Hideki Kawahara, Toshio Irino. 2005. The processing and perception of size information in speech sounds. *The Journal of the Acoustical Society of America* **117**:1, 305. [CrossRef]
13. Zhe Chen , Simon Haykin . 2002. On Different Facets of Regularization TheoryOn Different Facets of Regularization Theory. *Neural Computation* **14**:12, 2791-2846. [Abstract] [PDF] [PDF Plus]
14. Aki Vehtari , Jouko Lampinen . 2002. Bayesian Model Assessment and Comparison Using Cross-Validation Predictive DensitiesBayesian Model Assessment and

Comparison Using Cross-Validation Predictive Densities. *Neural Computation* **14**:10, 2439-2468. [Abstract] [PDF] [PDF Plus]

15. Randall C. O'Reilly . 2001. Generalization in Interactive Networks: The Benefits of Inhibitory Competition and Hebbian LearningGeneralization in Interactive Networks: The Benefits of Inhibitory Competition and Hebbian Learning. *Neural Computation* **13**:6, 1199-1241. [Abstract] [PDF] [PDF Plus]
16. M. Koppen, D.H. Wolpert, W.G. Macready. 2001. Remarks on a recent paper on the "no free lunch" theorems. *IEEE Transactions on Evolutionary Computation* **5**:3, 295-296. [CrossRef]
17. N.S.V. Rao. 2001. On fusers that perform better than best sensor. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **23**:8, 904-909. [CrossRef]
18. Malik Magdon-Ismail . 2000. No Free Lunch for Noise PredictionNo Free Lunch for Noise Prediction. *Neural Computation* **12**:3, 547-564. [Abstract] [PDF] [PDF Plus]
19. David Greenhalgh, Stephen Marshall. 2000. Convergence Criteria for Genetic Algorithms. *SIAM Journal on Computing* **30**:1, 269. [CrossRef]
20. Zehra Cataltepe , Yaser S. Abu-Mostafa , Malik Magdon-Ismail . 1999. No Free Lunch for Early StoppingNo Free Lunch for Early Stopping. *Neural Computation* **11**:4, 995-1009. [Abstract] [PDF] [PDF Plus]
21. Huaiyu Zhu , Wolfgang Kinzel . 1998. Antipredictable Sequences: Harder to Predict Than Random SequencesAntipredictable Sequences: Harder to Predict Than Random Sequences. *Neural Computation* **10**:8, 2219-2230. [Abstract] [PDF] [PDF Plus]
22. David H. Wolpert. 1997. On Bias Plus VarianceOn Bias Plus Variance. *Neural Computation* **9**:6, 1211-1243. [Abstract] [PDF] [PDF Plus]
23. D.H. Wolpert, W.G. Macready. 1997. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation* **1**:1, 67-82. [CrossRef]
24. David H WolpertBayesian and Computational Learning Theory . [CrossRef]