Natarajan, B. (1991). *Machine Learning: A theoretical approach.* Morgan Kauffman, 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., et al. (1994). Cross-validation estimates integrated mean squared error. In *Advances in neural information processing systems 6,* Cowan et al. (Ed.'s), Morgan Kauffman, CA.

Schaffer, C., (1993). Overfitting avoidance as bias. *Machine Learning*, **10**, 153-178.

Schapire, R. (1990). The strength of weak learnability. *Machine Learning.* **5**, 197-227.

Vapnik, V. (1982). Estimation of dependences based on empirical data. Springer-Verlag.

Vapnik, V., and Bottou, L. (1993). Local algorithms for pattern recognition and dependencies estimation. *Neural Computation,* **5**, 893-909.

Waller, W., Jain, A. (1978). On the monotonicity of the performance of Bayesian classifiers. *IEEE Transactions on Information Theory,* **IT-24**, 392-394.

Weiss, S.M., and Kulikowski, C. A. (1991). *Computer systems that learn*. Morgan Kauffman.

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

Wolpert, D. (1993). On overfitting avoidance as bias. SFI TR 93-03-016.

Wolpert, D. (1994a). The relationship between PAC, the Statistical Physics framework, the Bayesian framework, and the VC framework. To appear in *The future of supervised learning.* D. Wolpert (Ed.). Addison-Wesley.

Wolpert, D. (1994b). On the Bayesian "Occam factors" argument for Occam's razor. To appear in *Computational Learning Theory and Natural Learning Systems: Volume III Natural Learning Systems,* S. Hanson et al. (Ed.'s). MIT Press.

Wolpert, D. and Macready, W. (1994). No Free Lunch Theorems for Search. In preparation.

one $d_X(i) > k$. Now $E(C_{OTS} \mid f, d_X) \leq (n - k - 1) / (n - m) < (n - k) / (n - m)$.

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

References

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

Berger, J. (1985). *Statistical decision theory and Bayesian analysis.* Springer-Verlag.

Bernardo, J. Smith, A. (1994). *Bayesian Theory*. Wiley and Sons, NY.

Berger, J., and Jeffreys, W. (1992). Ockham's razor and Bayesian analysis. *American Scientist*, **80**, 64-72.

Blumer, A., et alia (1987). Occam's razor. *Information Processing Letters*, **24**, 377-380.

Blumer, A., et alia (1989). Learnability and Vapnik-Chervonenkis dimension. *Journal of the ACM*, **36**, 929-965.

Bridle, J. (1989). Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In F. Fougelman-Soulie and J. Herault (Eds.), *Neuro-computing: Algorithms, architectures, and applications.* Springer-Verlag.

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-Kauffman.

Duda, R., and Hart, P. (1973). *Pattern classification and scene analysis.* Wiley and Sons.

Hughes, G. (1968). On the mean accuracy of statistical pattern recognizers. *IEEE Transactions on Information Theory*, **IT-14**, 55-63.

Knill, M, Grossman, T., and Wolpert, D. (1994). Off-Training-Set Error for the Gibbs and the Bayes Optimal Generalizers. Submitted.

Mitchell T., Blum, A. (1994). *Course notes for Machine Learning*, CMU.

Murphy, P., Pazzani, M. (1994). Exploring the decision forest: an empirical investigation of Occam's razor in decision tree induction. *Journal of Artificial Intelligence Research*, **1**, 257-275.

In particular, for zero-one loss, and single-valued f, $s_{min}(d_X, \phi) = 0$, and $s_{max}(d_X, \phi) = \pi(d_X)$. So we need an h^* such $\{\varepsilon - \pi(d_X)\} / \pi(X - d_X) < E(C \mid \phi, h^*, d) \leq \varepsilon / \pi(X - d_X)$. For any $\varepsilon < 1$, there is such an h^*. This establishes theorem (12).

APPENDIX E. PROOF OF THEOREM (16)

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

E.1) $E(C_{OTS} \mid f, (no)\ punt, m) = \Sigma_{d_X} E(C_{OTS} \mid f, d_X) P(d_X \mid (no)\ punt, f, m)$

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

It's also true that $E(C_{OTS} \mid f, d_X)$ is not drastically different if one considers d_X's with a different m'. Accordingly, our summand doesn't vary drastically between d_X's of one m' and d_X's of another. Since n >> m though, almost all of the terms in the sum have m' = m (relatively speaking). 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.2) $E(C_{OTS} \mid f, (no)\ punt, m) = \Sigma_{d_X} E(C_{OTS} \mid f, d_X) P(d_X \mid (no)\ punt, m' = m)$.

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

D.2) $E(C \mid h_1, \phi, d) \leq \{\varepsilon - s_1\} / \pi(X - d_X)$,

and similarly for $E(C \mid h_2, \phi, d)$. (The expression "$\pi(\Xi)$" is shorthand for $\Sigma_{x \in \Xi} \pi(x)$.)

We are interested in $\Sigma_{h_1, h_2 \in H_\phi(\varepsilon)} E(C \mid h_1, h_2, \phi, d, i)$, where 'i' is either A or B. As usual, expand the sum into an outer and inner sum, getting

$\Sigma_{h_1(x \in d_X), h_2(x \in d_X)} \Sigma_{h_1(x \notin d_X), h_2(x \notin d_X)} E(C \mid h_1, h_2, \phi, d, i)$,

where the restriction that both h_1 and h_2 lie in $H_\phi(\varepsilon)$ is implicit.

Consider the inner sum, for any particular set of values of $h_1(x \in d_X)$ and $h_2(x \in d_X)$. Without loss of generality, say that for the d at hand and the values of $h_1(x \in d_X)$ and $h_2(x \in d_X)$, strategy A picks h_1. This means that $s_1 \leq s_2$.

Examine the case where s_1 is strictly less than s_2. Using (D.2), the inner sum over $h_1(x \notin d_X)$ is over all $h(x \notin d_X)$ such that $E(C \mid \phi, h, d) \leq \{\varepsilon - s_1\} / \pi(X - d_X)$, and the inner sum over $h_2(x \notin d_X)$ is over all $h_2(x \notin d_X)$ such that $E(C \mid \phi, h, d) \leq \{\varepsilon - s_2\} / \pi(X - d_X)$.

Since $s_1 < s_2$, this means that the h's going into the sum over $h_1(x \notin d_X)$ are a proper superset of the h's going into the sum over $h_2(x \notin d_X)$. In addition, for all the h's that are in the h_1 sum but not in the h_2, $E(C \mid \phi, h, d) \geq 0$. (In fact, those h's obey $\{\varepsilon - s_2\} / \pi(X - d_X) < E(C \mid \phi, h, d) \leq \{\varepsilon - s_1\} / \pi(X - d_X)$.) Therefore so long as the set $H^*(s_1, s_2)$ of h's in the h_1 sum but not in the h_2 sum is not empty, the h_1 sum is larger than the h_2 sum.

This means that for all cases where s_1 is strictly less than s_2 and $H^*(s_1, s_2)$ is non-empty, $\Sigma_{h_1(x \notin d_X), h_2(x \notin \in d_X)} E(C \mid h_1, h_2, \phi, d, i)$ is larger for algorithm A than for algorithm B. If $s_1 = s_2$, then $\Sigma_{h_1(x \notin d_X), h_2(x \notin \in d_X)} E(C \mid h_1, h_2, \phi, d, i)$ is the same for both algorithms. So if there are any h_1 and h_2 in the sum such that the associated $H^*(s_1, s_2)$ is non-empty, it follows that $\Sigma_{h_1, h_2 \in H_f(\varepsilon)} E(C \mid h_1, h_2, \phi, d, i)$ is larger for algorithm A than for algorithm B. If there are no such h_1 and h_2, $\Sigma_{h_1, h_2 \in H_f(\varepsilon)} E(C \mid h_1, h_2, \phi, d, i)$ is the same for both algorithms. This establishes theorem (11).

To understand when there are any h_1 and h_2 such that the associated $H^*(s_1, s_2)$ is non-empty (so that we get a strict inequality in theorem (11)), make the definition $s_{min}(\Xi, \phi) \equiv \Sigma_{q \in \Xi} \pi(q) \min_{y_h}(L(y_h, \phi(q)))$, and similarly for $s_{max}(\Xi, \phi)$. Then what we need is for there to be an h, h^*, such that

$\{\varepsilon - s_{max}(d_X, \phi)\} / \pi(X - d_X) < E(C \mid \phi, h^*, d) \leq \{\varepsilon - s_{min}(d_X, \phi)\} / \pi(X - d_X)$.

Note that the N(.) introduced here is the noise process operating in the generation of the test set, and need not be the same as the noise 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, $\phi(.)$, but to measure error by how well the resulting h matches that underlying $\phi(.)$, not by how well Y_h values sampled from h match Y_f values formed by adding noise to $\phi(.)$. In the ϕ-N terminology, this would mean that although $P(d \mid f)$ may be formed by corrupting some function $\phi(.)$ with noise (in either X and/or Y), $P(y_f \mid f, q)$, which measures test set error, is determined by a noise-free N(.), $N(y_f \mid q, \phi(q)) = \delta(y_f, \phi(q))$.

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 doesn't exclude q-dependent noise processes). I will call such a (test set) noise process "homogenous". 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 homogenous, as is a noise process that if it takes in a value of $\phi(q)$, produces the same value with probability z and all other values with (identical) probabilities $(1 - z) / (r - 1)$.

• 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.

APPENDIX D. PROOF OF THEOREMS (11) AND (12)

Let $H_\phi(\varepsilon)$ indicate the set of h's such that $E(C_{iid} \mid h, \phi) \leq \varepsilon$. This appendix analyzes the sign of $\Sigma_{h_1, h_2 \in H_\phi(\varepsilon)} [E(C \mid h_1, h_2, \phi, d, A) - E(C \mid h_1, h_2, \phi, d, B)]$ for OTS error. In particular, it shows that this quantity is necessarily positive for zero-one loss for any $\varepsilon < 1$.

We can express $H_\phi(\varepsilon)$ as the set of h's such that

D.1) $\Sigma_{q \in d_X} L(h(q), \phi(q)) \pi(q) + \Sigma_{q \notin d_X} L(h(q), \phi(q)) \pi(q) \quad \leq \quad \varepsilon.$

It is useful to introduce the following notation. Define $S(h, \phi, \Xi) \equiv \Sigma_{q \in \Xi} L(h(q), \phi(q)) \pi(q)$. Indicate a value of $S(h_1, \phi, d_X)$ as s_1, and a value of $S(h_2, \phi, d_X)$ as s_2. By (D.1), the set of h_1 under consideration are those for which $\varepsilon \geq s_1$, and such that for OTS error C,

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) = E(C | f, h, d), i.e., P(c' | f, h, d) = δ(c', E(C | f, h, d)).

Now 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 | d) need not give P(c' | d) or vice-versa.[11]. However many branches of supervised learning theory (e.g., much of computational learning theory) are concerned with quantities of the form “P(error > ε | ...)”.[12] For such quantities, whether one takes “error” to mean C or (as is conventional) C' may change the results, and in general one can not directly infer the result for C from that for C' (or vice versa).

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 only work out the results explicitly 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,

E(C | d) = $\int$ dhdf E(C | f, h, d) P(f, h | d) = $\int$ dhdf C'(f, h, d) P(f, h | d) = $\int$ dhdf E(C' | f, h, d) P(f, h | d) = E(C' | d).

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

• 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 X to Y with a fixed noise process in Y superimposed.[13] To do this, I will (perhaps only implicitly) fix a noise function N that is a probability distribution over Y, conditioned on X $\times$ Y; N is a probability distribution over y_f, conditioned on the values of q and ϕ(q). (Note that there are r^n such functions ϕ(.).)

Given N(.), each ϕ specifies a unique f via P(y_f | f, q) = f(q, y_f) = N(y_f | q, ϕ(q)) = P(y_f | q, ϕ). Accordingly, all the usual rules concerning f apply as well to ϕ. For example, P(h | d, ϕ) = P(h | d). When I wish to make clear what ϕ sets f, I will write f_ϕ; ϕ simply serves as an index on f. (In general, depending on N(.), it might be that more than one ϕ labels the same f, but this won’t be important for the current analysis.) So when I say something like “vertical P(d | ϕ)” it is implicitly understood that I mean vertical P(d | f_ϕ).

By “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.

same d always gives the same h (i.e., if for fixed d P(h | d) is a delta function about one particular h).

• In general, in supervised learning P(h | f, d) = P(h | d) (i.e., the learning algorithm only sees d in making its guess, not f). Similarly, P(f | h, d) = P(f | d), and therefore P(h, f | d) = P(h | d) P(f | d).

It follows that (for example) $P(y_f \mid y_h, d, q) = P(y_f \mid d, q)$. It does not follow 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.)

• There are many similar equalities that are assumed in this paper, but that do not merit explicit delineation. For example, it is implicitly assumed that P(h | q, d) = P(h | d), and therefore that P(q | d, h) = P(q | d).

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

• The random variable cost C is defined by $C = L(Y_h, Y_f)$, where L(., .) is called a "loss function". As examples, zero-one loss has $L(a, b) = 1 - \delta_{a,b}$, and quadratic loss has $L(a, b) = (a - b)^2$. Zero-one loss is assumed in almost all of computational learning theory.

For many L's 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 L's "homogenous". Intuitively, such L's have no *a priori* preference for one Y value over another. As examples, the zero-one loss is homogenous. So is the squared difference between two angles, $L(\theta_1, \theta_2) = [(\theta_1 - \theta_2) \bmod \pi]^2$. Note that one can talk of an L's being homogenous for certain values of c. For example, the quadratic loss isn't homogenous 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 homogenous for that c, even if L isn't homogenous for all c.

• Note that $E(C \mid f, h, d) = \Sigma_{y_h, y_f, q}\ E(C \mid f, h, d, y_h, y_f, q)\ P(y_h, y_f, q \mid f, h, d)$. 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 $h(q_h)\ f(q, y_f)\ P(q \mid f, h, d)$. In addition, $P(q \mid f, h, d) = P(q \mid d_X)$ always in this paper. Therefore

$$E(C \mid f, h, d) = \Sigma_{y_h, y_f, q}\ L(y_h, y_f)\ h(q, y_h)\ f(q, y_f)\ P(q \mid d).$$

• Note that f is a distribution governing test set data, and in general 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 by f.

• The value d of the training set random variable is an ordered set of m input-output examples. Those examples are indicated by $d_X(i)$, $d_Y(i)$ {i = 1 .. m}. The set of all input values in d is d_X and similarly for d_Y. m' is the number of distinct values in d_X.

• Often when training and testing sets are generated by the same P(y | x), the training set is formed by iterating the following "independent identically distributed" (iid) procedure: Choose X values according to a "sampling distribution" $\pi(x)$, and then sample f at the those points.[9] More formally, this very common scheme is equivalent to the following "likelihood":

$$(2.1) \qquad P(d \mid f) = P(d_Y \mid f, d_X)\, P(d_X \mid f) = P(d_Y \mid f, d_X)\, P(d_X) \text{ (by assumption)}$$
$$= \Pi_{i=1}^{m} [\, \pi(d_X(i))\, f(d_X(i), d_Y(i)) \,].$$

There is no *a priori* reason for P(d | f) to have this form however. For example, in "active learning" or "query-based" learning, successive values (as i increases) of $d_X(i)$ are determined by the preceding values of $d_X(i)$ and $d_Y(i)$. As another example, typically P(d | f) will not obey equation (2.1) if testing and training are not governed by the same P(y | x). To see this, let t be the random variable P(y | x) governing the generation of training sets. Then $P(d \mid f) = \int dt\, P(d \mid t)\, P(t \mid f)$. Even if $P(d \mid t) = \Pi_{i=1}^{m} [\, \pi(d_X(i))\, t(d_X(i), d_Y(i)) \,]$, unless P(t | f) is a delta function about t = f, P(d | f) need have the form specified in equation (2.1).

I will say that P(d | f) is "vertical" if it is independent of the values of $f(x \notin d_X)$. Any likelihood of the form given in equation (2.1) is vertical, by inspection. In addition, as discussed in section 6 below, active learning usually has a vertical likelihood. However some scenarios in which $t \neq f$ do not have vertical likelihoods.[10]

• The "posterior" is the Bayesian inverse of the likelihood, P(f | d). The phrase "the prior" usually refers to P(f).

• Any (!) learning algorithm (or "generalizer") is simply a distribution P(h | d). It is "deterministic" if the

(a particular hypothesis). This is standard statistics notation. (Note that with conditioning bars, this notation leads to expressions like "$P_{A|B}(a \mid b)$", meaning the probability of random variable A conditioned on variable B, evaluated at values a and b, respectively.)

Also in accord with standard statistics notation, "E(A | 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, $E(A \mid b) = \int dc\ da\ a\ P(a \mid b, c)\ P(c \mid b) = \int dc\ E(a \mid b, c)\ P(c \mid b)$. Finally, when it is obvious that their value is assumed fixed and what it is fixed to, sometimes I won't 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. (This is the only case in this paper where capital letters don't indicate random variables.) Such cases of countable X and Y are the simplest to present, and always obtain in the real world where data is measured with finite precision instruments and manipulated on finite size computers. A generic X value is indicated by x, and a generic Y value by y.

• In this paper, the "true" or "target" relationship between (test set) x's and (test set) y's is taken an X-conditioned distribution over Y (i.e., a $P(y \mid x)$). A special case of such a distribution is single-valued function from X to Y. Similarly, the generalizer's hypothesis is taken to be an X-conditioned distribution over Y. 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.

It will be useful to relate C to F, H, and D using three other random variables. "Testing" (involved in determining the value of C) is done at the X value given by the random variable Q. Y values associated with the hypothesis and Q are denoted by Y_H, and Y values associated with the target and Q are denoted by Y_F. All of this is formalized as follows.

• The random variables formed by sampling a target distribution F at the test point Q is Y_f. The variable Y_h is defined similarly. So for example, I will write $P(y_f \mid f, q) = f(q, y_f)$ and $P(y_h \mid h, q) = h(q, y_h)$.

Both f and h are sets of r^n real numbers lying on the n-fold Cartesian product of unit simplices. If for each q $h(q, y_h)$ is a delta function (over y_h), 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. (Note that in schemes like softmax [Bridle 1989] the system produced by the learning algorithm is a single-valued h, despite its being interpreted as a probability distribution; such schemes work by interpreting Y itself as a space of probability distributions.)

Write $P(c \mid y_h, q, d) = \int df \, \Sigma_{y_f} \delta(c, L(y_h, y_f)) \, f(q, y_f) \, P(f \mid y_h, q, d)$. Next use Bayes' rule to rewrite $P(f \mid y_h, q, d)$ as $P(y_h \mid f, q, d) \, P(f \mid q, d) / P(y_h \mid q, d)$.

Now write $P(y_h \mid q, d) = \Sigma_h \, h(q, y_h) \, P(h \mid d)$. But $P(y_h \mid f, q, d)$ is given by the same sum. Therefore $P(f \mid y_h, q, d) = P(f \mid q, d)$. Using Bayes' theorem again, $P(f \mid q, d) = P(q \mid d, f) \, P(d, f) \; / \; [\, P(q \mid d) \, P(d) \,] = P(d \mid f) \, P(f) / P(d) \propto P(d \mid f) / P(d)$ (since we have a uniform $P(f)$). So combining everything, up to an overall proportionality constant,

$$P(c \mid y_h, q, d) = \Sigma_{y_f} \delta(c, L(y_h, y_f)) \int df \; f(q, y_f) \, P(d \mid f) / P(d).$$

Now recall that $P(c \mid y_h, q, d)$ is occurring after being multiplied $P(q, d \mid y_h)$. For OTS error, $P(q, d \mid y_h) = 0$ unless $q \in X - d_X$. Therefore in evaluating $P(c \mid y_h, q, d)$ we can assume that $q \in X - d_X$. Accordingly, the term $f(q, y_f)$ only depends on the values of $f(x \notin d_X)$. Since we have a vertical likelihood, the $P(d \mid f)$ term only depends on $f(x \in d_X)$. Accordingly, we can write

$$P(c \mid y_h, q, d) = \Sigma_{y_f} \delta(c, L(y_h, y_f)) \int df(x \notin d_X) \; f(q, y_f) \; \int df(x \in d_X) \, P(d \mid f) \; / P(d).$$

Up to overall proportionality constants, we can now replace both integrals with $\int df$. By the same reasoning using in the proof of theorem (1), the first integral is a constant. If we reintroduce the constant $P(f)$ into the remaining integral, we get $\int df \, P(d \mid f) \, P(f) / P(d) = 1$. Therefore, $P(c \mid y_h, q, d) = \Sigma_{y_f} \delta(c, L(y_h, y_f))$. (Note that for homogenous $L(., .)$, this is independent of y_h.) As needed, this is independent of q and d. QED.

APPENDIX C. 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.

In general, unless indicated otherwise, capital letters are used to indicate random variables, and corresponding lower case letters to indicate a particular value of a random variable. When clarity is needed, the argument of a P(.) 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

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 doesn't change the functional form of the constraint forcing α to lie on a simplex. (I.e., $\Sigma_{\phi_1\phi_2''} \alpha_{\phi_1\phi_2''} = 1$ 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 $\int d\alpha \{ \alpha_{\phi_1,\phi_2^*} / \Sigma_{\phi_1'} \alpha_{\phi_1'} P(d \mid \phi_1') \}$, 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

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

for some function "$\text{func}_1(.)$". In other words, the α-average of $P(c \mid d, \alpha)$ is proportional to $\Sigma_{\phi_2} P(c \mid \phi_2, d)$, 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 $\Sigma_{\phi_1\phi_2} P(c \mid \phi, d) = \Sigma_{\phi} P(c \mid \phi, d)$. 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 "$\text{func}_2(.)$". Since $\Sigma_c P(c \mid d, \alpha) = 1$, 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 B. Proof of theorem 13

We need to prove that for a vertical likelihood and OTS error, for uniform P(f), all learning algorithms with the same $P(y_h \mid m)$ have the same $P(c \mid m)$. First write $P(c \mid m) = \Sigma_{y_h} P(c \mid y_h, m) \quad P(y_h \mid m)$ (where from now on the uniformity of P(f) is implicit). If $P(c \mid y_h, m)$ is independent of the learning algorithm, the proof will be complete. So expand $P(c \mid y_h, m) = \Sigma_{q,d} P(q, d \mid y_h) P(c \mid y_h, q, d)$. I will show that $P(c \mid y_h, q, d)$ is a fixed function of c and y_h that is independent of q and d. This in turn means that $P(c \mid y_h, m)$ is the same fixed function of c and y_h, which is exactly the result we need.

some d, we know that P(f) is 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 relationships between h and f for the off-training set q were possible, with equal probabilities. And in that second case, we would have the possibility of C' values that are impossible in the first case. QED.

12. 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.

13. Noise in X of the form mentioned in footnote 10 will not be considered in this paper. The extension to analyze such noise processes is fairly straight-forward however.

APPENDIX A. 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) $\int d\alpha [\Sigma_\phi P(\phi \mid d, \alpha) P(c \mid \phi, d, \alpha)]$, where as usual the integral is restricted to the r^n-dimensional simplex. Rewrite this integral as $\int d\alpha [\Sigma_\phi P(\phi \mid \alpha) P(c \mid \phi, d, \alpha) P(d \mid \phi, \alpha) / P(d \mid \alpha)] = \int d\alpha [\Sigma_\phi \alpha_\phi P(c \mid \phi, d) P(d \mid \phi)] / [\Sigma_{\phi'} \alpha_{\phi'} P(d \mid \phi')]$, where ϕ' is a dummy ϕ value. Rewrite this in turn as

$$\Sigma_\phi P(c \mid \phi, d) P(d \mid \phi) \quad \{ \int d\alpha \frac{\alpha_\phi}{\Sigma_{\phi'} \alpha_{\phi'} P(d \mid \phi')} \}.$$

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 X values lying inside d_X, and ϕ_2 fixes it over the values outside of d_X. We must find how the terms in our sum depend on ϕ_1 and ϕ_2.

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

behavior comes from guessing halfway between the lower and upper limits of Y.

Similarly, $E((y_h)^2 \mid m) = Var((y_h)^2 \mid m) + (E(y_h \mid m))^2$. So for two learning algorithms A and B with the same $E(y_h \mid m)$, guessing the algorithm with the smaller variance is preferable. These results justify the claims made at the beginning of this section, for the case of a uniform prior.

8. 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 doesn't directly address whether the presence of a punt signal gives you reason to believe you're in class (1), and therefore will have bad error. The explanation of why there is no such correlation is more subtle than simply counting the number of f's in each class. It involves the fact that there are actually a continuum of classes, and that for fixed f, raising s (so as to get a punt signal) lowers OTS (!) error.

9. 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 we make the usual assumption in the neural net and computational learning literature that π is fixed.

10. As an example, assume that f is some single-valued function from X to 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 X and noise in 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 X and Y values from the training set as random variables X_t and Y_t, one might have $Y_t = \Sigma_{X'} \gamma(X_t, X') \phi(X') + \varepsilon$, where X' is a dummy X variable, $\gamma(., .)$ is a convolutional process giving noise in X, and ε is a noise process in Y. (Strictly speaking, this particular kind of Y-noise makes sense if $r = \infty$, as otherwise $\Sigma_{x'} \gamma(x, x') \phi(x') + \varepsilon$ might not lie in Y.)

For this scenario, $t \neq f$. In addition, P(d | f) doesn't have the form given in equation (2.1). In particular, due to the convolution term, P(d | f) will depend on the values of $f = \phi$ for $x \notin d_X$; the likelihood for this scenario is not vertical.

11. If there are only two possible L(., .) values (for example), P(c' | d) does give P(c | d). This is because P(c' | d) gives E(C' | d) = E(C | d) (see below), and since there are two possible costs, E(C | d) gives P(c | d). It is for more than two possible cost values that P(c' | d) and P(c | d) do not set one another. In fact, even if there are only two possible values of L(., .), so that P(c' | d) sets P(c | d), it does not follow that P(c | d) sets P(c' | d). As an example, consider this case where n = m' + 2, and we have zero-one loss. Assume that given

4. In terms of appendix D, we still get $s_1 < s_2$. In appendix D this meant that the upper limit on h_2's error over X - d_X was less than the upper limit on h_1's error, while they shared lower limits. Here it instead means that the lower limit on h_2's error over X - d_X is lower than the lower limit on h_1's error, while they share the same upper limit.

5. Phrased differently, for a quadratic loss function, given a series of experiments and a set of deterministic learning algorithms G_i, it is always preferable to use the average generalizer $G' \equiv \Sigma_i G_i / \Sigma_i 1$ for all such experiments rather than to randomly choose a new member of the G_i to use for each experiment. Intuitively, this is because such an average reduces variance without changing bias. (See [Perrrone 1993] for a discussion of what is essentially the same phenomenon, in a neural net context.) Note though that this in no way implies that using G' for all the experiments is better than using any single particular $G \in \{G_i\}$ for all the experiments.

6. To see this, write $P(y_h \mid y_f, m) = \Sigma_{d,q} \int dfdh\, h(q, y_h) P(h \mid d) P(d, q, f \mid y_f, m)$. Use Bayes' theorem to write $P(d, q, f \mid y_f, m) = P(y_f \mid f, d, q) P(f, d, q \mid m) / P(y_f \mid m) = f(q, y_f) P(q \mid d) P(d \mid f) P(f) \ / \ P(y_f \mid m)$. Combining, $P(y_h \mid y_f, m) = \Sigma_{d,q} \int dfdh\, h(q, y_h) P(h \mid d) f(q, y_f) P(q \mid d) P(d \mid f) P(f) \ / \ P(y_f \mid m)$.

In the usual way, we take P(f) to be a constant, have the P(q | d) restrict q to lie outside of d_X, and then break the integral over f into two integrals, one over the values of $f(x \in d_X)$, and one over the values of $f(x \notin d_X)$. The integral over $f(x \notin d_X)$ transforms the $f(q, y_f)$ term into a constant, independent of q and y_f. We can then bring back in the P(f), and replace the integral over $f(x \in d_X)$ with an integral over all f (up to overall proportionality constants). So up to proportionality constants, we're left with

$P(y_h \mid y_f, m) = \Sigma_{d,q} \int dh\, h(q, y_h) P(h \mid d) P(q \mid d) P(d) / \ P(y_f \mid m)$.

Now write $P(y_f \mid m) = \Sigma_{d,q} \int df\, P(y_f \mid f, d, q) P(f, d, q \mid m) \propto \Sigma_{d,q} \int df\, f(q, y_f) P(q \mid d) P(d \mid f)$. As before, the P(q | d) allows us to break the integral over all f into two integrals, giving us some overall constant. So $P(y_h \mid y_f, m) \propto \Sigma_{d,q} \int dh\, h(q, y_h) P(h \mid d) P(q \mid d) P(d)$. However this is exactly the expression we get if we write out $P(y_h \mid m)$. QED.

7. As an aside, consider the case where Y = {1, 2, ..., r} and r is odd. Now $E((y_h)^2 \mid m) - 2 E(y_h \mid m) E(y_f \mid m) = -[E(y_f \mid m)]^2 + E((y_h - E(y_f \mid m))^2 \mid m)$. So guessing such that y_h always equals $E(y_f \mid m)$ gives best behavior. In particular, for the likelihood of equation (2.1), $E(y_f \mid m) = \Sigma_{i=1}^{r} i / r = (r + 1) / 2$, so best

8) Investigate under what circumstances $E(C_{iid} \mid m) < E(C_{OTS} \mid m)$.

9) Find the set of P(f)'s such that all learning algorithms are equivalent, as determined by P(c | m). (There are other P(f)'s besides the uniform one for which this is so. E.g., for fixed noise, the distribution that's uniform over ϕ's.) Find the set of P(f)'s for which two particular algorithms are equivalent.

10) Investigate the s-conditioned distributions and punt-signal-conditioned distributions mentioned as "beyond the scope of this paper" in section 6.

11) Investigate if and how results change if one conditions on a value of m' rather than m;

12) Carry through the analysis for error C' rather than C.

Acknowledgments: I would like to thank Cullen Schaffer, Wray Buntine, Manny Knill, Tal Grossman, Bob Holte, Tom Dietterich and Jeff Jackson for interesting discussions. This work was supported in part by the Santa Fe Institute and by TXN Inc.

FOOTNOTES

1. 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 "minimax" ideas that touch on this paradox is presented below.

2. For more than two possible values of L(.), it's not clear what happens. Nor is it clear how much of this carries over to costs C' (see section (2)) rather than C.

3. Even if minimax behavior does end up playing favorites between algorithms, it's not clear why one should care about such behavior rather than simple expected cost (which plays no such favorites). One intriguing possible answer to this question is that in choosing between species A and species B (and their associated organic learning algorithms), natural selection may use minimax behavior. The idea is that for those f's for which A's behavior is just slightly preferred over B's, equilibrium has a slightly smaller population for species A than for species B. But if there is any non-linearity in the system, then for any f's for which A's behavior is far worse than B's, A goes extinct. Therefore if over time the environment presents A and B with a series of f's, the surviving species will be the one with preferable minimax behavior.

ample, for the conventional loss function, noise process etc. studied in computational learning theory, there are indeed no generalization assurances associated with averages. (As a result, if we know that (for example) averaged over some set of targets F the generalizer CART is superior to some canonical neural net scheme, then we also know that averaged over targets not contained in F, the neural net scheme is superior to CART.) On the other hand, there may be some assurances associated with non-averaging criteria, like minimax criteria. In addition, for certain other noise processes and/or certain loss functions one *can*, a priori, say that one algorithm is superior to another, even in terms of averages.

More generally, for all its reasonableness when stated in the abstract, the full implications of the no-assurances concept for applied supervised learning can be surprising. For example, it implies that cross-validation fails as readily as it succeeds, boosting makes things worse as readily as better, active learning and/or algorithms that can choose not to make a guess fail as readily as they succeed, etc.

In addition to these kinds of issues, in which the generalizer is fixed and one is varying the target, this paper also discusses scenarios where the generalizer can vary but the target is fixed. For example, this paper uses such analysis to show that if one averages over generalizers cross-validation is no better than "anti" cross-validation, *regardless of the target*. In this sense, cross-validation can not be justified as a Bayesian procedure - no prior over targets justifies its use for all generalizers.

There are many avenues for future research on the topic of OTS error. Some of them are mentioned in [Knill et al. 1994]. Even for the limited nfl aspect of OTS error, there are still many issues to be explored. Some of them are:

1) Characterize when cross-validation beats anti-cross-validation for non-homogenous loss functions;

2) Investigate distributions of the form $P(c_1, c_2 \mid ...)$, where c_i is the error for algorithm i, when there are more than two possible values of the loss function;

3) Investigate the validity of the "minimax" justification for cross-validation, both for the scenario discussed in this paper, and for non-homogenous loss functions and/or s-conditioned distributions, and/or averaging over hypotheses rather than targets;

4) Investigate whether there are scenarios in which one generalizer is minimax-superior to the majority of other generalizers. Investigate whether there are "minimax superior cycles", in which A is superior to B is ... superior to A;

5) Investigate for what distributions over h's sums over h's with f fixed result in the majority algorithm beating the anti-majority algorithm;

6) Investigate sums over h's with f fixed when f and/or h is not single-valued (note for example that if $f(q, y) = 1 / r \; \forall$ q and y, then all h's have the same value of $\Sigma_{y_h, y_f} f(q, y_f) h(q, y_h) L(y_h, y_f)$ if L(., .) is homogenous);

7) Investigate sums over h's where f too is allowed to vary.

2) PAC (and indeed all of computational learning theory) has nothing to say about this-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 doesn't go to the limit, and hold even for this-data scenarios.

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) f's 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 don't emphasize the importance of a vertical likelihood; they don't emphasize the importance of homogenous noise when the target is a single-valued function; they don't emphasize the importance of whether the loss function is homogenous; they don't invoke scrambling for non-homogenous loss functions; they don't concern averaging over pairs of h's (in the sense of section (4)), etc.

5) Most of PAC (and much of computational learning theory) is based on assuming that there are some f with *exactly* zero prior probability (namely, the f outside of the "concept class" being analyzed). Such an f would be a theoretical impossibility; observing it would be tantamount to pulling down the pillars of heaven. Obviously this assumption is wholly incorrect. There is never a scenario in which some f is 100% impossible. In contrast, no obviously incorrect assumptions are made in the nfl theorems. In fact, one might argue that they make no assumptions at all; rather their purpose is to show that assumptions (hopefully valid ones!) must be made to justify use of a particular learning algorithm.

6) Computational learning theory does not address OTS error. Especially when m is not infinitesimal in comparison to n and/or $\pi(x)$ is highly non-uniform, 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 (2).)

7. CONCLUSIONS AND FUTURE WORK

This paper investigates some of the behavior of OTS (off-training set error). In particular, it formalizes and investigates the concept that "if you make no assumptions concerning the target, then you have no assurances about how well you generalize".

As stated in this general manner, this no-assurances concept is rather intuitive and many researchers would agree with it readily. However the details of what "no assurances" means are not so obvious. For ex-

i.e., P(no punt | α = uniform P(f), 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: P(f) = $\int d\alpha$ P(f | α) P(α) and is fixed beforehand, independent of the data. So the presence / absence of a punt signal can't be used to "infer" something about P(f), 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 α's rather than f's. 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(no punt | class, m) strongly favors class 2, the posterior does 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 an f with a non-negligible percentages of 1's, it's hugely unlikely to get all 0's in a large random data set. Hence, so this intuitive reasoning goes, if you get all 0's, you can conclude that f does not have a non-negligible percentages of 1's, and therefore you are safe in guessing 0's outside the training set. The contrasting intuition: say you are given some particular training set, say of the first K points in X, together with associated Y values. Say the Y values happen to be all 0's. Obviously, without some assumption concerning the coupling of f's behavior over the first K points in X with its behavior outside of those points, f could have any conceivable behavior outside of those points. So the fact that it is all 0's has no significance, and can not help you in guessing.

It should be emphasized that none of the reasoning of this subsection directly addresses the issue of whether the punting algorithm has good minimax OTS behavior in some sense (see the end of section (3)). 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.

vi) 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.

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 $E(C_{OTS} \mid m)$ is "good" (however defined). In lieu of such an analysis, one can find those P(f) such that $E(C_{OTS} \mid \text{no punt}, m) < E(C_{OTS} \mid \text{punt}, m)$. (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.

v) Intuitive arguments concerning the nfl theorems and punting algorithms

Consider again the algorithm addressed in theorem (16). For this algorithm, there are two kind of f:

1) f is such that the algorithm will almost always punt for a d sampled from f, or

2) f is such that the algorithm has tiny expected error when it chooses not to punt.

(Targets f with almost no x's such f(x) = 1 are in the second class, and other targets are in the first class.)

It might seem that this justifies use of the algorithm. After all, if the target is such that there's a non-negligible probability that the algorithm doesn't punt, it's not in class 1, and error will be tiny. So it would seem that *whatever the target* (or prior over targets), if the algorithm hasn't punted, we can be confident in its guess. (Similar arguments can be made when the two classes distinguish sets of P(f)'s rather than f's.)

However if we restate this, the claim is that E(C | no punt, m) is tiny for sufficiently large m, for any P(ϕ). (Note that for n >> m, m' is unlikely to be much less than m.) This would imply in particular that it is tiny for uniform P(ϕ). However from the preceding subsection we know that this isn't true. So we appear to have a paradox.

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

1) The target is such that 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 f's in class 1 than in class 2. So even though the probability of our no-punt signal is small for each of the f's in class 1 individually, when you multiply by the number of such f, you see that the probability of being in class 1, given that you have a no-punt signal, isn't 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're in, and therefore in determining likely error.[8]

So at a minimum, one must assume that P(f) is not uniform to have justification for believing the punt / no punt signal. Now one could argue that a uniform P(f) is highly unlikely when there's a no-punt signal,

$P(c \mid m, \text{no punt}) = \Lambda(c) / r$. 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 non-uniform 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 non-homogenous error.

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 $q \notin d_X$. 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 f, the probability of one such 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 non-negligible ε, $P(c_{OTS} > \varepsilon, \text{no punt} \mid f, m)$ is tiny.

However $P(c_{OTS} > \varepsilon, \text{no punt} \mid f, m)$ in fact equals 0 for the always-punt algorithm. So one might want to also consider other distributions like $P(c_{OTS} > \varepsilon \mid \text{no punt}, f, m)$ or $P(c_{OTS} < 1 - \varepsilon, \text{no punt} \mid f, m)$ 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 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 and targets are single-valued, and there is no noise. Y is binary, and we have zero-one loss. Let the learning algorithm always guesses the all 0's function, h*. The punt signal is given if d_Y contains at least one non-zero value. Then for the likelihood of (2.1), uniform $\pi(x)$, and $n >> m$, we have the following result, proven in appendix E:

Theorem 16 For all f such that $f(x) = 0$ for more than m distinct x, $E(C_{OTS} \mid f, \text{punt}, m) \leq E(C_{OTS} \mid f, \text{no punt}, m)$.

For $n >> m$, essentially all f meet the requirement given in theorem (16).

In many respects, the proper way to analyze punting algorithms is as follows. 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

between algorithms as far as P(c | d) is concerned, it is hard to see why one should choose between algorithms based on distributions of the form P(c | function(d), h), if one does indeed know d in full.

iii) 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 (2.1), 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.

Such active learning likelihoods are vertical. 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.

iv) 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, an 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 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* 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 | f), and a homogenous OTS error, P(c | d) is the same as that of a random generalizer, i.e., under those conditions, P(c | d, no punt) = Λ(c) / r. As a direct corollary,

$P(c'_{OTS} \mid m)$ and $P(s \mid m)$ were both tightly clumped around the same value.

Intuitively, many of the computational learning theory results relating s and c'_{iid} are driven by the fact that s is formed by sampling c'_{iid}. However for OTS c' s can not be viewed this way. Rather s and c'_{OTS} are on an equal footing. Indeed, for single-valued f's and h's with no noise, s and c'_{OTS} are both simply the value c'_{iid} has when restricted to a particular region in X. (The region is d_X for s, X - d_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.

ii) Implications of those nfl theorems that involve empirical error

The s-independence of the results of the preceding subsection 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 s is very low and 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 rule out a uniform P(f) *a priori*). 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 $P(c'_{OTS} \mid m)$ and $P(s \mid m)$ being peaked about the same value. (See [Wolpert 1994a] for further discussion of how independence of s and c'_{OTS} is compatible with the VC theorems.)

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 section (4).

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 | function(d), h), as opposed to distributions of the form P(c | d) (or perhaps P(c | d, h), or the f-average of P(c | d, f), or some such). So since the nfl theorems say there is no *a priori* distinction

and assume a noise-free iid likelihood. Then for any value of m', m's + (n - m')c'_{OTS} = constant. So $P(c'_{OTS} \mid s, f, m) = \delta(c, (\text{constant} - m's) / (n - m'))$, and gets biased towards higher values of c'_{OTS} as s shrinks. Now invoke the argument in footnote 11 to extend this result to c_{OTS}.

So we do not have nfl behavior for the uniform average over f of P(c | f, s, m). In fact, it is hard to say anything general about P(c | f, s, m). In particular, it is not always the case that higher s results in lower c if f is fixed, as in the example above. To see this, simply set up a scenario such that if all elements of d_X are in some region Ξ, then h = f, whereas for any other d_X's, there are errors both on and off d. So if s = 0, we know that $c_{OTS} = 0$. But if s > 0, we know that $c_{OTS} > 0$; raising s from 0 has raised expected C_{OTS}.

Now consider P(c | s, d) for uniform P(f). To evaluate this quantity 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 thought that P(s | f, d) = P(s | d) (see beginning of this section). 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 $f(x \in d_X)$, and one over $f(x \notin d_X)$. The result is the following theorem that holds for any learning algorithm.

Theorem 15 For homogenous L, OTS error, a vertical likelihood, and uniform P(f), $P(c \mid s, d) = \Lambda(c) / r$.

Often there are f's 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 $P(c_{OTS} \mid f, s, m)$ is not defined, although $P(c_{OTS} \mid s, m)$ for uniform P(f) is. Moreover, the set of f for which $P(c_{OTS} \mid f, s, m)$ is defined will vary with s. This explains how theorem (15) can hold despite the example above where as s shrinks $P(c_{OTS} \mid f, s, m)$ gets biased towards higher values of c_{OTS}, for any f for which both distributions (for both values of s) are defined.

As usual, an immediate corollary of theorem (15) is that under the conditions in theorem (15), $P(c \mid s, m) = \Lambda(c) / r$, independent of the learning algorithm. This in turn means that P(c | s, m) = P(c | m) under the conditions in theorem (15) (see corollary (1)). This implies that P(s | c, m) is independent of c. So C_{OTS} and S are statistically independent, for uniform P(f), homogenous L, and a vertical likelihood.

In accord with this, one expects that C'_{OTS} and S are independent for uniform P(f). On the other hand, VC theory tells us that for any P(f), P(c' - s | m) is biased towards small values of c' - s for low-VC dimension generalizers, large m, and iid c'. It should be emphasized that there is no contradiction in this. Indeed, such a bias in $P(c'_{iid} - s \mid m)$ could occur even if c'_{iid} and c'_{OTS} closely approximate each other. In other words, even if c'_{OTS} closely approximates c'_{iid}, independence of s and c'_{OTS} does not imply that that s and c' can differ significantly. For example, such biasing would occur if despite independence of s and c'_{OTS},

ample, 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). 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. Then it analyzes OTS behavior of "membership queries" algorithms and "punting" algorithms (those that may refuse to make a guess).

i) Nfl theorems involving empirical error

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 h's. For such learning algorithms, d determines 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 | f, d, s), for these kinds of learning algorithms.

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

Lemma 3 $P(c \mid f, d, s) = \Sigma_{y_h, y_f, q}\, \delta(c, L(y_h, y_f))\, P(y_h \mid q, d, s)\, P(y_f \mid q, f)\, P(q \mid d)$,

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 h's):

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

Unfortunately, one can not continue paralleling the analysis in section (2) 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 $P(d \mid f, s, m)$. In fact, it's not uncommon that for all f for which it is defined, $P(c_{OTS} \mid f, s, m)$ becomes more biased towards high values of c_{OTS} as s *shrinks*. Intuitively, for fixed single-valued f and no noise, a linear combination of c'_{OTS} and s gives c'_{iid}, so raising s can shrink c'_{OTS}. The same kind of effect means that raising s can shrink $E(C_{OTS} \mid f, s, m)$ even for non-single-valued f.

As an example of such effects, let $P(h \mid d) = \delta(h, h^*)$ for some h^*, let $\pi(x)$ be uniform, use zero-one loss,

As shown in example N, for this scenario, $E(C \mid m) = E((y_h - E(y_f \mid m))^2 \mid m)$. For our scenario, for a uniform P(f), $E(y_f \mid m) = 2$. So in comparing the uniform f-average of expected cost for cross-validation with that for anti-cross-validation, it suffices to compare the values of $E((y_h - 2)^2 \mid m)$ for the two techniques.

Now by symmetry, whatever d_X is, d_Y is equally likely to be 1, 2, or 3. Therefore if $d_X \in \{1, 2, ..., n / 2\}$, for cross-validation, there is a 2/3 chance of choosing h_1 (cross-validation will choose h_1 if $d_Y = 2$ or 3, for such a d_X). Similarly, for anti-cross-validation, there is a 2/3 chance of choosing h_2. Now since q must lie outside of d_X, for such a d_X, $E((y_h - 2)^2 \mid m)$ is smaller if y_h is given by h_2 than if it is given by h_1.

So anti-cross-validation does better if $d_X \in \{1, 2, ..., n / 2\}$. For analogous reasons, anti-cross-validation also does better if $d_X \in \{1 + n/2, ..., n\}$. So anti-cross-validation wins regardless of d_X. QED.

Example: As an example of where cross-validation beats anti-cross-validation, consider the same scenario as in the preceding example, only with different h_1 and h_2. Have $h_1 = 2$ for all x, and $h_2 = 1$ for all x. Cross-validation is more likely to guess h_1, and anti-cross-validation is more likely to guess h_2, regardless of d_X. However h_1 has a better $E((y_h - E(y_f \mid m))^2 \mid m)$ than does h_2, again, for all d_X. QED.

Note the important feature of the preceding two examples that whether cross-validation works (in comparison to anti-cross-validation) depends crucially on what learning algorithms you're using it to choose among. This is another illustration of the fact that assuming that cross-validation works is not simply making an assumption concerning the target, but rather making an assumption about the relationship between the target and the learning algorithms at hand.

Intuitively, if one uniformly averages over all h_1 and h_2, there is no statistical correlation between the training set and the cost, regardless of the learning algorithm. When the loss function is homogenous, this lack of correlation results in the nfl theorems. When the loss function isn't homogenous, it instead gives the results of this section.

6. 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 $s \equiv \Sigma^m_{i=1} \Sigma_{y_h} L(y_h, d_Y(i)) h(d_X(i), y_h) \pi(d_X(i)) \; / \; \Sigma^m_{i=1} \pi(d_X(i))$. As an ex-

Combining theorem (14) and corollary 3, we see that for the likelihood of equation (2.1) and OTS error, if algorithm B is a scrambled version of algorithm A, then the algorithms have the same uniform average over f of P(c | f, m).

Note that if $\pi(x)$ is uniform, we can relax the definition of scrambling to simply have Σ_d P(h | d, A) = Σ_d P(h | d, B) and theorem (14) will still hold. (For such a case P(q | d_X) will be a function purely of m' that can be absorbed into func(m, m').)

iii) Implications of the equivalence of any learning algorithm with its scrambled version

The results of the preceding subsection tell us that there is no *a priori* reason to believe that there is any value in a particular learning algorithm's relationship between d and h. All that matters about the algorithm is how prone it is to guess certain y_h's, not the conditions determining when it makes those guesses.

However these results are not as strong as the nfl theorems. As an example, these results do tell us that cross-validation (used to choose among a pre-fixed set of learning algorithms) and its scrambled version always gives the same uniform f-average of P(c | f, m) (if one has OTS error, etc.). So in that sense there is no *a priori* justification for the d-dependence of cross-validation, even for non-homogenous loss functions. However for non-homogenous loss functions we can not automatically equate cross-validation with anti-cross-validation, like we could for homogenous loss functions. This is because the two techniques can have different P(y_h | m). In fact, there are some scenarios in which cross-validation has a better uniform f-average of P(c | f, m), and some in which anti-cross-validation wins instead. Not only can't one equate the techniques, one can't even say that the relative superiority of the techniques is always the same.

Example: As an example of where anti-cross-validation has better uniform f-average of its expected cost than does cross-validation, let both techniques be used to choose between the pair of learning algorithms A and B. Let A be the algorithm "always guess h_1 regardless of the data" and B be the algorithm "always guess h_2 regardless of the data". Let the training set consist of a single element, and let the "validation set" part of the training set be that single element. So cross-validation will guess whichever of h_1 and h_2 is a better fit to the training set, and anti-cross-validation will guess whichever is a worse fit to the training set.

Let Y = {1, 2, 3}, and X = {1, 2, ..., n} where n is even. Let $h_1(x) = 2$ for $x \in \{1, 2, ..., n / 2\}$, and let it equal 1 for the remaining x. Conversely, $h_2(x) = 1$ for $x \in \{1, 2, ..., n / 2\}$, and 2 for the remaining x. Assume we are using the quadratic loss function to measure C, and that both cross-validation and anti-cross-validation use the quadratic loss function to choose between h_1 and h_2. Assume the likelihood of equation (2.1) and a uniform sampling distribution. Use a uniform P(f).

the learning algorithm through $P(y_h \mid m)$.[7]

Now $P(c \mid m) = \Sigma_f P(c \mid f, m) P(f \mid m)$. Therefore $P(c \mid m)$ for a uniform $P(f)$ is proportional to $\Sigma_f P(c \mid f, m)$, where the proportionality constant is independent of the learning algorithm. This establishes the following corollary of theorem (13).

Corollary 3 Consider two learning algorithms that would have the same $P(y_h \mid m)$ if $P(f)$ were uniform. For a vertical likelihood and OTS error, the uniform average over f of $P(c \mid f, m)$ is the same for those two algorithms.

Now make the following definition.

Definition 1 The learning algorithm B is a "scrambled" version of the learning algorithm A if and only if for all d_X, $\Sigma_{d_Y} P(h \mid d_X, d_Y, B) = \Sigma_{d_Y} P(h \mid d_X, d_Y, A)$. (The sums are implicitly restricted to d_Y's with the same number of elements as d_X.)

Intuitively, if B is a scrambled version of A, then B is identical to A except that A's correlation between its guess and the (output components of the) data has been scrambled.

As an example, view (a deterministic) algorithm A as a big table of h values, indexed by d's. Then a scrambled version of A is given by the same table where the entries within each block corresponding to a particular d_X have been permuted. Note that if for certain training sets A creates h's with low training set error, then in general a scrambled version of A will have much higher error on those training sets.

Theorem 14 Assume the (vertical) likelihood of equation (2.1). Then if learning algorithm B is a scrambled version of A, it follows that for uniform P(f), $P(y_h \mid B, m) = P(y_h \mid A. m)$.

Proof. For a uniform P(f), $P(y_h \mid m) \propto \Sigma_{q,d} \int df\, dh\, h(q, y_h) P(h \mid d) P(q \mid d) P(d \mid f)$. The integral over f only works on the $P(d \mid f)$. For the likelihood at hand, it is some function func(m, m'), independent of d. Therefore we have

$$P(y_h \mid m) \propto \Sigma_{q,d} \int dh \;\; func(m, m')\, h(q, y_h) P(h \mid d) P(q \mid d_X)$$

$$= \Sigma_{q,d_X} \int dh \;\; func(m, m')\, h(q, y_h) P(q \mid d_X) \Sigma_{d_Y} \;\; P(h \mid d_X, d_Y).$$

By hypothesis, the sum over d_Y is the same for our two algorithms. QED.

relationship between d and h is randomly scrambled. Such comparisons show that all the *a priori* advantages conferred on a particular learning algorithm by a non-homogenous L(., .) are due simply to the clumping of $P(y_h)$ in regions that, due to L(., .) are "good". (An example of such a region is the region equidistant from the borders of Y, for a quadratic L(., .).) In particular, according to these comparisons, there is no extra advantage conferred by how the learning algorithm chooses to associated y_h's with particular d's and q's. I.e., there is no advantage conferred by the d-dependence of the algorithm.

This "scrambling" tool can also be applied to non-homogenous noise-processes. For reasons of space though, such an analysis is not presented here.

ii) The equivalence of any learning algorithm with its scrambled version

For reasons of space, I will only consider the case where all f's are allowed (rather than the case where are f's expressible as some ϕ with noise superimposed are allowed). Since the focus is on how (if at all) the statistical relationship between d and h embodied in the generalizer correlates with error, we must allow d to vary. Accordingly, the results are conditioned on m rather than on d.

The following theorem, which holds regardless of the loss function, is proven in appendix B.

Theorem 13 For a vertical likelihood, uniform P(f), and OTS error, all learning algorithms with the same $P(y_h \mid m)$ have the same $P(c \mid m)$.

Example N: Say we have quadratic L(., .). In general, we will only be interested in the case where $r > 2$. (For binary Y, e.g., $Y = \{0, 1\}$, quadratic loss is the same as zero-one loss, and we're right back in the setting where the nfl theorems apply.) Rather than consider $P(c \mid m)$ directly, consider the functional of it, $E(C \mid m)$. For quadratic L(., .), $E(C \mid m) = E((y_h - y_f)^2 \mid m)$. Expanding, we get

$$E(C \mid m) = E((y_h)^2 \mid m) + E((y_f)^2 \mid m) - 2\, E(y_h\, y_f \mid m).$$

$E((y_f)^2 \mid m)$ is some constant, independent of the learning algorithm, determined by the geometry of Y, the likelihood, etc. $E((y_h)^2 \mid m)$ does depend on the learning algorithm, and is determined uniquely by $P(y_h \mid m)$, in accord with theorem (13).

The only term left to consider is the correlation term $E(y_h\, y_f \mid m)$. Write this term as $\int dy_f\, y_f\, E(y_h \mid y_f, m)\, P(y_f \mid m)$. Now for the assumptions going into theorem (13), $E(y_h \mid y_f, m) = E(y_h \mid m)$.[6] Accordingly, there is no expected correlation between y_h and y_f: $E(y_h\, y_f \mid m) = E(y_h \mid m)\, E(y_f \mid m)$. This means in turn that our correlation term only depends on the learning algorithm through $E(y_h \mid m)$, which in turn is uniquely determined by $P(y_h \mid m)$. Therefore the theorem is corroborated; $E(C \mid m)$ only depends on

we use the majority algorithm with on the other. In particular, it can't be that the only restriction on those pairs of h_1 and h_2 is that they lie in a sphere of radius ε centered on f. However it might be that certain "reasonable" non-uniform distributions over the set of allowed h's result in our empirical truth. Alternatively, there may be other geometrical restrictions (besides a sphere) that allow h_1 to be close to h_2, but that also allow our empirical truth. As one example, it is conceivable that our empirical truth is allowed if h_1 and h_2 don't fall in a sphere of radius ε centered on f, but rather in a ellipse centered off of f.

5. NON-HOMOGENOUS LOSS FUNCTIONS

i) Overview

This section considers the case where L(., .) is not homogenous, so that the nfl theorems given in section 3 do no apply, and therefore we usually can make *a priori* distinctions between algorithms. This section is not meant to be an exhaustive analysis of such loss functions, but rather to illustrate some of the properties and issues surrounding such functions.

An example of such an L(., .) is the quadratic loss function, $L(a, b) = (a - b)^2$ for finite Y. For the quadratic loss function (and in general for any convex loss function when Y is finite), everything else being equal, an algorithm whose guessed Y values lie away from the boundaries of Y is to be preferred over an algorithm which guesses near the boundaries. In addition, consider using a quadratic loss function with two learning algorithms, $P_1(h \mid d)$ and $P_2(h \mid d)$. Construct the algorithms to be related in the following manner: algorithm 2 is a deterministic algorithm that makes the single-valued guess given by the expected y_h guessed by algorithm 1, in response to the d and q at hand. (More formally, $P_2(h \mid d) = \delta(h - h^*)$, where $h^*(q, y) = \delta(y, E_1(y_h \mid q, d)) = \delta(y, \Sigma_{y'} [y' \int dh' \, h'(q, y') \, P_1(h' \mid d))]$), where h' is a dummy h argument and y' a dummy y argument.) It is not hard to show [Wolpert 1994a] that for all d and q, $E_2(C \mid q, d) \leq E_1(C \mid d, q)$, i.e., the expected cost for algorithm 2 is always less than or equal to that of algorithm 1, regardless of properties of the target.[5] This holds even for OTS error, so long as the loss function is quadratic.

None of this means that the intuition behind the nfl theorems is faulty for non-homogenous L(., .). However it does mean that that intuition now results in theorems that are not so sweeping as the nfl theorems. In essence, one must "mod out" the possible *a priori* advantages of one algorithm over another that are due simply to the fact that those algorithms tend to result in y_h values that are favored *a priori* by the L(., .) at hand.

The primary tool for deriving these less-sweeping results is to compare the OTS behavior of a learning algorithm P(h | d) to that of its "scrambled" or "randomized" version, in which $P(y_h)$ is preserved, but the

iii) Restricted averages over quantities involving the hypothesis

Since theorem (9) tells us that we must "match" h_1 and h_2 to f for the majority algorithm (A) to beat anti-majority, one might expect that if we sum only over those h_1 and h_2 lying close to f, then A beats B. Intuitively, if you have a pretty good idea of what f is, and restrict h's to be fairly good approximators of that f, then you might expect majority to beat anti-majority.

Interestingly, the exact opposite is usually true, if one measures how "close" an h is to f as $E(C_{iid} \mid f, h) = \Sigma_{y_h, y_f, q} L(h(q), y_f) f(q, y_f) \pi(q)$. The analysis of this is in appendix D, where in particular the following results are derived.

Theorem 11 For OTS error, any d, and any single-valued f, ϕ, if there is no noise in the generation of the training set, then for the majority and anti-majority strategies A and B,

$\Sigma_{h_1, h_2} [E(C \mid h_1, h_2, \phi, d, B) - E(C \mid h_1, h_2, \phi, d, A)] \leq 0,$

where the sum is over all h_1 and h_2 such that both $E(C_{iid} \mid f, h_1)$ and $E(C_{iid} \mid f, h_2)$ are less than ε.

Theorem 12 For zero-one OTS error, any d and any (single-valued) f, ϕ, if there is no noise in the generation of the training set, then for the majority and anti-majority strategies A and B,

$\Sigma_{h_1, h_2} [E(C \mid h_1, h_2, \phi, d, B) - E(C \mid h_1, h_2, \phi, d, A)] < 0,$

where the sum is over all h_1 and h_2 such that both $E(C_{iid} \mid f, h_1)$ and $E(C_{iid} \mid f, h_2)$ are less than 1.

As usual, since these results hold for any particular d, they also hold if one averages over d's generated from ϕ. In addition, similar results hold if A and B refer to cross-validation and anti-cross-validation rather than the majority strategy and the anti-majority strategy. (Rather than sum over all h_1 and h_2 in a certain class, one sums over all generalizers G_1 and G_2 that produce h's that lie in that class.)

The opposite results hold if we instead restrict h_1 to h_2 to lie far from ϕ, i.e., if we restrict attention to h's for which $E(C_{iid} \mid \phi, h) > \varepsilon$.[4] It is currently unknown what happens if we adopt other kinds of restrictions on the allowed h's. In particular, it is not clear what happens if one h must be in one region and the other h in a different region.

Say we accept it as empirically true that using the majority algorithm does, in the real world, usually result in lower expected cost than using the anti-majority algorithm, as far as OTS zero-one loss is concerned. Say we accept this as true even in those cases where we use h's that are similar, and therefore have similar goodness-of-fits to f. Given theorems (11) and (12), this implies that there must be some rather subtle relationship between the f's we encounter in the real world on the one hand, and the pairs of h_1 and h_2 that

Theorem 10 For an OTS homogenous symmetric error, the uniform average over all generalizers G_1, G_2 of P(c | f, G_1, G_2, d, cross-validation) equals the uniform average over all G_1 and G_2 of P(c | f, G_1, G_2, d, anti-cross-validation).

So for any f, averaged over all generalizers, cross-validation does the same as anti-cross-validation.

All of this holds even if we're considering more than two h's at a time (in the case of theorem (9)), or more than two generalizers at a time (in the case of theorem (10)).

ii) Implications of theorems (9) and (10)

Since they hold for all f, theorems (9) and (10) mean that there is no prior P(f) such that, without regard to h_1 and h_2 (G_1 and G_2 in the case of theorem (10)), strategy A is preferable to strategy B, for either of the two sets of strategies considered in those theorems. (Note how much stronger this statement is then saying that averaged over all P(f), two strategies are equal.) In this sense, there is no Bayesian justification for strategy A; B beats A just as readily as vice-versa. At best, strategy A beats strategy B for certain P(f) *and certain associated h_1 and h_2.* Exactly which h_1 and h_2 result in the superiority of strategy A will change with P(f).

So a technique like cross-validation can not be justified by making assumptions only concerning P(f), nor by making assumptions only concerning G_1 and G_2. Rather one must make assumptions about how G_1 and G_2 correspond to P(f). It is the interplay between all three quantities that determines how well cross-validation performs. (See theorem (1) of section (3) of [Wolpert 1994a].)

Since these results hold for all d, they mean in particular that if f is fixed while d is averaged over, then the two strategies have equal average OTS error. This is important because such a scenario of having f fixed and averaging over d's is exactly where you might presume (due to computational learning theory results) that strategy A beats strategy B.

How can we reconcile the results of this section with those computational learning theory results? Consider the case of theorem (9). My suspicion is what's happening is the following: There are relatively few $\{h_1, h_2\}$ pairs for which strategy A wins, but for those pairs, it wins by a lot. There are many pairs for which strategy B wins, but for those pairs, strategy B only wins by a little. In particular, I suspect that those "many pairs" are the relatively many pairs for which f agrees with h_1 almost exactly as often as it agrees with h_2.

If this is indeed the case, it means that strategy A is unlikely to be grossly in error. Note that this is a confidence-interval statement, exactly the kind that the VC theorems apply to. (However it is a confidence-interval statement concerning *off-training set* error; in this it is actually an extension of VC results, which concern iid error.)

$\Sigma_{h,q,y_h,y_f} \delta(c, L(y_h, y_f)) P(q \mid d) f(q, y_f) h(q)$.

We can use the reasoning of the nfl theorems to calculate this. Intuitively, all we need do is note by lemma (1) that our sum equals $\int df\, P(c \mid f, d)$ if one interchanges f and h, and then invoke theorem (1). More formally, follow the reasoning presented after lemma (1), only applying it to our distribution rather than to $\int df\, P(c \mid f, d)$. The result is lemma (2) (rather than theorem (1)). QED.

Now consider the quantity $\Sigma_{h_1,h_2} P(c \mid f, h_1, h_2, d, i)$, where the variable i is either A or B, depending on which strategy we're using. This quantity tells us whether A or B is preferable, if we uniformly average over all pairs of h's one might use with A and/or B. (Non-uniform averages are considered below.) As such, it is a measure of whether A is preferable to B or vice versa.

Expand our sum as

$$\Sigma_{h_1(x \in d_X), h_2(x \in d_X)} \Sigma_{h_1(x \notin d_X), h_2(x \notin d_X)} P(c \mid f, h_1, h_2, d, i).$$

Consider any set of values of $h_1(x \in d_X)$, $h_2(x \in d_X)$ such that the strategy at hand picks h_1. For such a set of values, our inner sum becomes (up to an overall proportionality constant determined by m') $\Sigma_{h_1(x \notin d_X)} P(c \mid f, h_1, d)$, with obvious notation. By lemma (2), this is simply some function $\lambda(c)$, independent of f and d. The same is true for those sets of values of $h_1(x \in d_X), h_2(x \in d_X)$ such that the strategy at hand picks h_2. Therefore, up to overall constants, $\Sigma_{h_1,h_2} P(c_i \mid f, h_1, h_2, d)$ is just $\lambda(c)$. This is true regardless of which strategy we use. We have now established the following.

Theorem 9 For an OTS homogenous symmetric error, $\Sigma_{h_1,h_2} P(c \mid f, h_1, h_2, d, A) = \Sigma_{h_1,h_2} P(c \mid f, h_1, h_2, d, B)$, for all f and d.

So even if the likelihood is not vertical, averaged over all h_1 and h_2, strategy (A) equals strategy (B).

The same reasoning can be used to equate cross-validation with anti-cross-validation. Let G_1 and G_2 be two learning algorithms, and let strategies A and B now refer to cross-validation and anti-cross-validation respectively. Consider uniformly averaging $P(c \mid f, G_1, G_2, d, i)$ over all G_1 and G_2. Since d is fixed, this corresponds to uniformly averaging over all possible hypotheses h_1 and h_2 constructed from d, and an independent average over all possibilities of whether it is the hypothesis labelled h_1 or the one labelled h_2 that corresponds to the generalizer with lower cross-validation error. Consider just the inner-most of these two sums, i.e., without loss of generality say that it is h_1 that corresponds to strategy A and h_2 that corresponds to strategy B. Then by lemma 2, we deduce the following.

4. AVERAGING OVER GENERALIZERS RATHER THAN TARGETS

In all of the discussion up to this point, we have averaged over entities concerning targets (namely f, ϕ, or P(ϕ)) while leaving entities concerning the hypothesis (e.g., the generalizer P(h | d)) fixed. Although the results of such an analysis are illuminating, it would be nice to have alternative results in which one doesn't have to specify a distribution over f/ϕ/P(ϕ). Such results would be prior-independent.

One way to do this is to hold one of f/ϕ/P(ϕ) fixed (though arbitrary), and average over entities concerning the hypothesis instead. It is not clear if one can formulate this approach in so sweeping a manner as the nfl theorems, but even its less sweeping formulation results in some interesting conclusions. In that they are prior-independent "negative" conclusions, these conclusions constitute a first principles proof that, for example, cross-validation is non-Bayesian. I.e., they constitute a proof that there is no f/ϕ/P(ϕ) for which cross-validation will necessarily work, independent of the learning algorithms it's choosing among.

i) Averages over entities concerning the hypothesis

Consider the following situation: f is fixed, as is the training set d. There are two hypothesis functions, h_1 and h_2. For simplicity, I will restrict attention to the (most popular) case where h's are single-valued. Consider two strategies for choosing one of the h_i:

A) Choose whichever of the h_i agrees more often with d;

B) Choose whichever of the h_i agrees less often with d.

Note that if the h's agree with d just as often as each other, the strategies are equivalent.

To analyze the relative merits of strategies A and B, start with the following lemma.

Lemma 2 For all f, for all d having m' distinct elements, and for all sets of m' values for $h(x \in d_X)$

$\Sigma_{h(x \notin d_X)} P(c \mid f, h, d)$

is the same function of c, for an OTS homogenous loss that is a symmetric function of its arguments.

Proof: Expand the sum in question as

$\Sigma_{h(x \notin d_X),q,y_h,y_f} \delta(c, L(y_h, y_f)) P(q \mid d) f(q, y_f) h(q)$.

Since $P(q \mid d) = 0$ for $q \in d_X$, this sum is independent of the values of $h(x \in d_X)$. Accordingly, up to an overall constant, we can rewrite it as

perior to the algorithms in that set.

This leads to the question of whether one can construct a fixed, learning algorithm that *is* minimax superior to all others. (As a possible alternative, it may be that there are "cycles", in which algorithm A is minimax superior to B, and B is to C, but in addition C is superior to A.) It seems unlikely that the answer to this question is yes. After all, due to the nfl theorems, the smallest $\max_f E(C \mid f, m)$ can be for our candidate learning algorithm is Σ_c c $\Lambda(c)$ / r. However for all f, there is at least one learning algorithm that has zero error on that f (the algorithm that guesses that f regardless of the data set). It is hard to reconcile these facts with our hypothesis, especially given that the Σ_c is usually quite large (e.g., for zero-one loss, it's (r - 1) / r).

It may be that if one restricts attention to "reasonable" algorithms (presumably defined to exclude the always-guess-f-regardless-of-d algorithm, among others), there is such a thing as an algorithm that is minimax superior to all others. This raises the following interesting issue: All learning algorithms that people use (which, in a bout of circular logic, might constitute almost all that people would consider "reasonable") are very similar to one another. To give a simple example, almost all such algorithms try to fit the data, but also restrict the "complexity" of the hypothesis somehow. So the picture that emerges is that people use learning algorithms tightly clustered in a tiny section of the space of all algorithms. It may be that there is an algorithm that is minimax superior to all those in the tiny cluster, even if there is no such thing as a universally minimax superior algorithm. If that is the case, then it would be possible to construct an algorithm "superior" to those currently known, but only because people have had such limited imagination in constructing learning algorithms.

vii) Other peculiar properties associated with OTS error

There are many other aspects of OTS error which, although not actually nfl theorems, can nonetheless be surprising. An example is that in certain situations the expected 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 error on T is meaningless. In this, even the ubiquitous use of test sets is unjustified, unless one makes assumptions concerning the target. For a discussion of this point and of intuitive arguments for why the nfl theorems hold, see [Wolpert 1994a].

being examined) A and B have approximately the same expected OTS error, but β usually chooses the worse of the two, so in such situations the expected cost of β is (slightly) worse than that of α. In those comparatively few situations where A and B have significantly different expected OTS error, β might correctly choose between them, so the expected cost of β is significantly better than that of α for such situations. In other words, it might commonly be the case that when asked to choose between two generalizers in a situation where they have comparable expected cost, cross-validation usually fails, but in those situations where the generalizers have significantly different costs, cross-validation successfully chooses the better of the two.

Similar behavior may hold even when using cross-validation to choose among more than two algorithms. In such a case, cross-validation still has the same average OTS behavior as any other algorithm. And there are actually more situations in which it fails than in which it succeeds (!). However in such a case cross-validation has desirable minimax behavior; its behavior will never be much worse than the best possible.[3]

Note that such desirable minimax behavior would be prior independent; if it holds for all f, then it certainly holds for all P(f). In addition, such behavior would be consistent with the (conventional) view that cross-validation is usually an accurate estimator of generalization performance. It's important to note though that one can explicitly construct cases where cross-validation doesn't have this desirable minimax behavior. (See section 8 in [Wolpert 1994a].)

There are a number of other interesting hypotheses related to minimax behavior. For example, it may be that in many situations not only is cross-validation minimax superior to anti-cross-validation, but also minimax superior to the generalizers it's choosing among. So in particular, assume we are in such a scenario, and that whatever f is, the generalizers amongst which we are choosing all perform well for that f. (I.e., we believe our generalizers are well-suited to f, although we perhaps can not formalize this in terms of priors over f, etc.) Then we are assured that cross-validation will not perform much worse than the best of those generalizers, and may perform significantly better. Due to the nfl theorems, this would mean that for most f cross-validation performs slightly worse than the generalizers it's choosing among. However even if the f at hand is such an f, since by hypothesis our generalizers perform well for that f, cross-validation will perform well for that f. (It should be noted though that even if this hypothesis holds, there are still a number of caveats concerning the use of cross-validation, caveats that, unlike the nfl theorems but just like minimax behavior, apply regardless of f. See the averaging-over-hypotheses section below.)

In the preceding hypothesis, the technique of cross-validation constitutes a different learning algorithm for each new set of generalizers it chooses among. So even if the hypothesis is true, it would not mean that there is a single learning algorithm that is minimax superior to all others. Rather it would mean that given any (presumably not too large) set of learning algorithms, one can construct a new one that is minimax su-

tities are addressed in the nfl theorems.

As a special case of the theorems, when there are only two possible values of L(., .), any two algorithms are even more tightly matched in behavior than theorems (1) through (8) indicate. (An example of such an L(., .) is zero-one loss, for which there are only two possible values of L(., .) regardless of r.) Let C_1 and C_2 be the costs associated with two learning algorithms. Now $P(c_1 \mid \text{stuff}) = \Sigma_{c_2} P(c_1, c_2 \mid \text{stuff})$, and similarly for $P(c_2 \mid \text{stuff})$. (Examples of "stuff" are {d, f}, {m}, etc.) If L(., .) 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). Accordingly, if we know both $P(c_1 \mid \text{stuff})$ and $P(c_2 \mid \text{stuff})$, we can solve for $P(c_1, c_2 \mid \text{stuff})$ (sometimes up to some overall unspecified parameters, since our four 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(., .) 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).[2]

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_X and $\pi(x \in d_X)$. 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.

Notwithstanding the nfl theorems though, learning algorithms can differ in that: 1) for particular f, or particular (non-uniform) P(f), different algorithms 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.

vi) The nfl theorems and minimax behavior

It is interesting to speculate about the possible implications of point (3) of the previous subsection for cross-validation. Consider two algorithms α and β. α is identical to some algorithm A, and β works by using cross-validation to choose between A and some other algorithm B. By the nfl theorems, α and β must have the same expected error, on average. However the following might hold for many choices of A, B, $\pi(x)$, etc.: For most situations (i.e., most f, or most ϕ, or most $P(\phi)$, depending on which of nfl theorem's averages is

sumptions about the applicability of that "knowledge". This is because that knowledge is ultimately based on only two things: your experiences since birth, and your genome's experiences in the several billion years it's 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, and the nfl theorems apply. In particular, every time you use induction to infer how to deal with some novel aspect of the modern world, you are dealing with something that none of your "prior knowledge" directly concerns. By the nfl theorems therefore, you have no formal justification for the use you make of your prior knowledge in such a situation.

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 assurances about its behavior in the future.[1]

It should also be emphasized that one can not simply say, for example, "okay, cross-validation corresponds to some assumption or other about targets; I make that assumption", and thereby formally justify one's use of cross-validation. To date, no one has written down what that assumption is; you can't formally invoke an unknown assumption. (In fact, it is shown below that there is no assumption solely concerning targets that justifies cross-validation in all contexts.)

Finally, it is important to emphasize that results based on averaging uniformly over f/ϕ/P(ϕ) shouldn't 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 non-homogenous 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*.

v) General implications of the nfl theorems

The primary importance of the nfl theorems is their implication that, according to any of the distributions P(c | d), P(c | m), P(c | f, d) or P(c | f, m), all algorithms are equivalent, on average. Or to put it another way, for any two learning algorithms, there are just as many situations (appropriately weighted) in which algorithm one is superior to algorithm two as vice-versa, according to any of the measures of "superiority" addressed in the nfl theorems. 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.

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 | f, m) (or other quantities determined by P(c | f, m). (See [Wolpert 1994a].) Similarly, conventional Bayesian analysis is concerned with P(c | d). All of these quan-

useful tool.

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(ϕ) 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 is usually superior to the second. The nfl theorems show that this is not the case. If all f's are weighted by their associated probability of error, then for any two algorithms A and B there are exactly as many f's for which A beats B as vice-versa.

Now if one changes the weighting over f's 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) not uniform, so the nfl results don't apply to my favorite learning algorithm" are misguided at best. Unless you can prove that the non-uniformity in P(f) is well-matched to your favorite learning algorithm (rather than being "anti-matched" to it), the fact that P(f) may be non-uniform, by itself, provides no justification whatsoever for your use of that learning algorithm. (See theorem (1) in [Wolpert 1994a].)

In fact, the nfl theorems for averages over priors P(ϕ) 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 (!).

Given all this, 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, this was the basis for the "problem-averaging" work of Hughes [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 other words, it would be nice if, in an investigation of first principles distinctions between algorithms, one could justify some other kind of averaging besides uniform averaging. However it is extremely difficult to see how such a justification could be arrived at. It seems that either one can not investigate these kinds of issues, or one must use uniform averaging.

In this regard, note that you really would need a proof based completely on first principles to formally justify some particular non-uniform P(f). In particular, you can not 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 as-

so (expected) C = .5 for both algorithms. Therefore for each of these ϕ's, expected C = .4(1) + .6(.5) = .7 for algorithm A, and .4(0) + .6(.5) = .3 for B.

iv) The case of ϕ's with two 1's is the same as the case of ϕ's with three 1's (just with '1' replaced by '0' throughout). Similarly, one 1 = four 1's, and zero 1's = five 1's. So it suffices to just consider the cases already investigated, where the number of 1's is zero, one, or two.

v) Adding them up, for algorithm A we have one ϕ with (expected) C = 0, five with C = .2 and 10 with C = .7. So averaged over all those ϕ's, we get [1(0) + 5(.2) + 10(.7)] / [1 + 5 + 10] = .5. This is exactly the same expected error as algorithm B has: expected error for B is [1(1) + 5(.8) + 10(.3)] / 16 = .5. QED.

Example 2) An algorithm that uses cross-validation to choose amongst a pre-fixed set of learning algorithms does no better on average than one that doesn't, so long as the loss function is homogenous. 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-homogenous loss (e.g., quadratic loss), and this result will still hold; all that is required is that we use a homogenous loss to measure error on the test set.

Note that this result doesn'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].

Example 3) Assume you're a Bayesian, and calculate the Bayes-optimal guess assuming a particular P(f). (I.e., you use the P(h | d) that would minimize the data-conditioned risk E(C | 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 one 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, etc. The nfl theorems say that all such heuristics fail as often (appropriately weighted) as they succeed.

iv) On uniform averaging

The uniform sums over f (or ϕ, or P(ϕ)) in the nfl theorems weren't chosen because there is strong reason to believe that all f are equally likely to arise in practice. Rather they were chosen because such a sum is a

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

Theorem 8 Assume OTS error, a vertical $P(d \mid \phi)$, homogenous L, and a homogenous 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 $\Lambda(c) / r$.

It may be that for the (homogenous) noise process at hand, for some d's and ϕ's, $P(c \mid \phi, d)$ is not defined. This is the situation for example when there is no noise (d must lie on ϕ). In such a situation, averaging over all ϕ's 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 save 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.

iii) Examples

Example 1: Say we have no noise, and the zero-one L(.). 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 "anti-majority" 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., $Y = \{0, 1\}$) and a uniform $\pi(x)$. Take m' = 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 1's h, and h_2 = the all 0's h.

i) There is one ϕ that is all 0's (i.e., for which for all X values, Y = 0). For that ϕ, for algorithm A $E(C \mid \phi, m' = 4) = 0$; algorithm A performs perfectly. For algorithm B, expected C = 1.

ii) There are five ϕ's with one 1. For each such ϕ, the probability that the training set has all four zeroes is .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 A, and 1 for B. So for each of these ϕ's, the expected value of C is .2(1) + .8(0) = .2 for A, and .2(0) + .8(1) = .8 for B.

iii) There are ten ϕ's with two 1's. For each such ϕ, there is a .4 probability that the training set has one 1, and a .6 probability that it has both 1's. (It can't have no 1's.) 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 1's, then our algorithms say guess randomly,

Theorem 5 For OTS error, a vertical $P(d \mid \phi)$, homogenous L, and a homogenous test-set noise process, the uniform average over all 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 $P(d \mid \phi)$, homogenous L, uniform $P(\phi)$, and a homogenous 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 $P(d \mid \phi)$, homogenous L, a homogenous 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 $P(\phi)$ rather than over 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) $\int d\alpha \, P(c \mid m, \alpha) = \int d\alpha \, [\Sigma_\phi \, P(\phi \mid m, \alpha) \, P(c \mid m, \alpha, \phi)]$, where the integral is restricted to the r^n-dimensional simplex. (α is restricted to lie on that simplex, since $\Sigma_\phi \, P(\phi \mid \alpha) = \Sigma_\phi \, \alpha_\phi = 1$.)

This can be rewritten as $\int d\alpha \, [\Sigma_\phi \, \alpha_\phi \, P(c \mid \phi, m, \alpha)]$, since we assume that the choice of ϕ has nothing to do with the choice of the number of elements in d. Similarly, once ϕ is fixed, the probability that $C = c$ doesn't depend on α, so our average equals $\int d\alpha \, [\Sigma_\phi \, \alpha_\phi \, P(c \mid \phi, m)]$. Write this as $\Sigma_\phi \, P(c \mid \phi, m) \, [\int d\alpha \, \alpha_\phi]$. 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 $\Sigma_\phi \, P(c \mid \phi, m)$. Using theorem (5) and normalization, we have now established the following:

Theorem 7 Assume OTS error, a vertical $P(d \mid \phi)$, homogenous L, and a homogenous 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 $\Lambda(c) / r$.

for OTS error, homogenous L(., .), and a generalizer such that P(d | h) is independent of h($x \notin d_X$), the uniform average over h of P(c | h, m) = Λ(c) / r. (For such a generalizer, assuming $h_1(x) = h_2(x)$ for all $x \in d_X$, 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 h's the algorithm might produce, all posteriors (and therefore all priors) are the same, under the specified conditions.

ii) Averaging over all functions ϕ

Now consider the scenario where only those f are allowed that can be viewed as single-valued functions ϕ from X to Y with noise superimposed (see section 2). 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 f's that are single-valued ϕ's with noise superimposed.)

In this new scenario, lemma (1) still holds, with f replaced by ϕ. However now we can not simply set the uniform ϕ-average of P(y_f | q, ϕ) to 1 / r. To give an extreme example, if the test set noise process is highly skewed and sends all ϕ(q) to some fixed value y_1, then the ϕ-average is 1 if $y_f = y_1$, 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_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 | q, ϕ) is independent of y_f. Recall that such a (test set) noise process is called "homogenous". If we sum our ϕ-average of P(y_f | q, ϕ) 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. Accordingly, the ϕ-average equals 1 / r. So we have the following analog of theorem (1):

Theorem 4 For homogenous L and a homogenous test-set noise process, the uniform average over all functions ϕ of P(c | ϕ, d) equals Λ(c) / r.

Note that the noise process in generating the training set is irrelevant to this result. (Recall that "homogenous noise" refers to y_f and y_h, and that y_f and y_h are 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, if our error measure is concerned with whether or not h equals the underlying ϕ at the point q (a measure corresponding to noise-free - and therefore homogenous - test set generation).

We can proceed from theorem (4) to get a result for P(c | f, m) in the exact same manner as theorem (1) gave theorem (2). The result is

P(c | f, d) (lemma (1)) with a sum over those q lying outside of d_X. Accordingly, for such a P(q | d), P(c | f, d) is independent of the values of $f(x \in d_X)$. (For clarity, the second argument of f is being temporarily suppressed.)

Assume further that P(d | f) is vertical (i.e., independent of the values of $f(x \notin d_X)$). Next average both sides of our equation for P(c | f, m) uniformly over all f and pull the f-average inside the sum over d. Since P(c | f, d) and P(d | 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

$$\Sigma_d \; [\int df(x \notin d_X) \; P(c \mid f, d) \;\; \int df(x \in d_X) \; P(d \mid f)] \;\; / \;\; \int df(x \notin d_X) \; df(x \in d_X) \; 1 \; .$$

But since P(c | 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 For OTS error, a vertical P(d | f), and a homogenous L, the uniform average over all f of $P(c \mid f, m) = \Lambda(c) / r$.

Again, this holds for any learning algorithm, and any sampling distribution.

Now write P(c | 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 $f(x \in d_X)$ 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 $\int df(x \notin d_X) \; 1$. So we have the following theorem concerning the distribution of interest in conventional Bayesian analysis, P(c | d):

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

As an immediate corollary we have the following.

Corollary 1. For OTS error, a vertical P(d | f), uniform P(f), and a homogenous 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 EBF is symmetric under interchange of h and f. (In particular, for any L, P(f | h, d) = P(f | d), P(h | f, d) = P(h | d).) Accordingly, all of the nfl theorems have analogues where h rather than f is fixed and then uniformly averaged over. So for example,

$(1 / r)\ \Sigma_{y_h, y_f, q}\ \delta(c, L(y_h, y_f))\ P(y_h \mid q, d)\ P(q \mid d)$.

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

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

Note that this f-average is independent of the generalizer. So theorem (1) constitutes an nfl theorem for distributions conditioned on f and 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 zero-one loss, we get the f-average of $E(C \mid f, d) = 1 / r$. More generally, for an even broader set of L's than homogenous L's, the sum over y_f of $L(y_h, y_f)$ is independent of y_h. For such L's we get generalizer-independence for the uniform average over f of E(C | f, d), even if we don't have such independence for the uniform average of P(c | f, d).

Note that theorem (1) doesn't 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) isn't 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 f and d. As the first of these, I will show how to use theorem (1) to evaluate the uniform f-average of P(c | f, m) for OTS error.

In evaluating the uniform f-average of P(c | f, m), not all f's contribute the same amount to the answer. That's because

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

and the P(d | f) term will favor those f's for which d's giving large P(c | f, d) are more likely. This might lead one to suspect that if the learning algorithm is "biased" towards the f contributing the most to the uniform f-average of P(c | f, m) this would weight the average towards low values of c. However this is wrong; it turns out that the uniform f-average of P(c | f, m) is independent of the learning algorithm, if one restricts oneself to OTS error.

In fact, assume that we have any P(q | d) such that $P(q \in d_X \mid d) = 0$ (in particular, P(q | d) need not be the OTS P(q | d) discussed above). For such a scenario, we can replace the sum over all q that gives

First I derive the nfl theorems for the case where f need not be single-valued. In that case, the theorems say that uniformly averaged over all f, all learning algorithms are identical.

When f is not single-valued, it is a (countable) set of real numbers (one for each possible x-y pair). Accordingly, P(f) is a probability density function in a multi-dimensional space. That makes integrating over P(f)'s a subtle mathematical exercise. However in the function+noise scenario "f" is indexed by a single-valued function ϕ. Since there are a countable number of ϕ's, P(ϕ) is a countable set of real numbers, and it is straight-forward to integrate over all P(ϕ). 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 f.

Here and throughout this paper, when discussing non-single-valued f's, "A(f) uniformly averaged over all 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). Similar meanings for "uniformly averaged" are assumed if we're talking about averaging over other quantities, like P(ϕ).

i) Averaging over all distributions f

We start with the following lemma.

Lemma 1 $P(c \mid f, d) = \Sigma_{y_h, y_f, q}\ \delta(c, L(y_h, y_f))\ P(y_h \mid q, d)\ P(y_f \mid q, f)\ P(q \mid d)$.

Proof. Write $P(c \mid f, d) = \Sigma_{y_h, y_f, q}\ P(c \mid y_h, y_f, q, f, d)\ P(y_h \mid y_f, q, f, d)\ P(y_f, q \mid f, d)$. Re-writing the summand, we get $P(c \mid f, d) = \Sigma_{y_h, y_f, q}\ \delta(c, L(y_h, y_f))\ P(y_h \mid y_f, f, q, d)\ P(y_f, q \mid f, d)$.

Now $P(y_h \mid y_f, f, q, d) = \Sigma_h\ P(y_h \mid y_f, h, f, q, d)\ P(h \mid y_f, f, q, d) = \Sigma_h\ P(y_h \mid h, q, d)\ P(h \mid q, d)$ (see point (10) in the previous 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 [Knill et al. 1994].) Plugging in gives the result. QED.

Now uniformly average P(c | f, d) over all f. The only place f occurs in the sum in lemma (1) is in the third term. 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 q and y_f. Accordingly $\text{func}(y_f, q)$ is some constant. Since the sum over y_f of this constant must equal 1 (interchange the sum over y_f with the average over f), that constant must equal 1 / r. Accordingly,

The uniform average over all f of P(c | f, d) equals

(2.1) $$P(d \mid f) = \Pi_{i=1}^{m} \pi(d_X(i)) \; f(d_X(i), d_Y(i))$$

(where $\pi(x)$ is the "sampling distribution") 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 | f). (See appendix C.)

9) The "posterior" usually means P(f | d), and the "prior" usually means P(f).

10) The learning algorithm only sees d, so P(h | f, d) = P(h | d), P(f | h, d) = P(f | d), and therefore P(h, f | d) = P(h | d) P(f | d).

11) The cost c associated with a particular y_h and y_f is given by the loss function $L(y_h, y_f)$. L(., .) is "homogenous" if the sum over y_f of $\delta(c, L(y_h, y_f))$ is some function $\Lambda(c)$, independent of y_h. As an example, the "zero-one" loss traditional in computational learning theory (L(a, b) = 1 if $a \neq b$, 0 otherwise) is homogenous.

12) In the case of "iid error" (the conventional error measure), P(q | d) = $\pi(q)$ (so test set inputs are chosen according to the same distribution that sets training set inputs). In the case of OTS error, P(q | d) = [$\delta(q \notin d_X) \, \pi(q)$] / [Σ_q numerator], where $\delta(z) \equiv 1$ if z is true, 0 otherwise.

13) The "generalization error function" used in much of supervised learning is given by $C' \equiv E(C \mid f, h, d)$. It is the average over all q of C, for a given f, h, and d. In general, probability distributions over C' do not set those over C nor vice-versa. However the results in this paper in general hold for both C and C', although they will only be presented 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.

14) It will be convenient at times to restrict attention to f's that are constructed by adding noise to a single-valued function from X to Y, ϕ. Such f's are indexed by the underlying ϕ. The noise process is "homogenous" if the sum over all ϕ of $P(y_f \mid q, \phi)$ is independent of y_f.

3. THE NO-FREE-LUNCH THEOREMS

ferent random variable for the hypothesis output by the learning algorithm and for the target relationship. (It is this distinction that separates the EBF from conventional Bayesian analysis.) This section presents a synopsis of the EBF. Readers unsure of any aspects of this synopsis are directed to the detailed exposition of the EBF in appendix C.

1) Aside from the input and output spaces X and Y, capital letters indicate random variables, and lower case letters indicate instantiations of a random variable. In accord with standard statistics notation, "E(A | b)" will be used to mean the expectation value of A given B = b, i.e., to mean $\int$ da a P(a | b). (Sums replace integrals if appropriate.)

2) n and r are the number of elements in X and Y respectively.

3) The primary random variables are the hypothesis X-Y relationship output by the learning algorithm (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 the (test set) input space value (q), and the associated target and hypothesis Y-values, y_f and y_h respectively.

4) m is the number of elements in the (ordered) training set d. $\{d_X(i), d_Y(i)\}$ is the set of m input and output values in d. m' is the number of distinct values in d_X.

5) Hypotheses h are always assumed to be of the form of X-conditioned distributions over Y, indicated by $h(x \in X, y \in Y)$ (i.e., $P(y_h \mid h, q) = h(q, y_h)$). 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 function from X to Y.

6) Any (!) learning algorithm (aka "generalizer") is given by P(h | d). It is "deterministic" if the same d always gives the same h.

7) Targets f are always assumed to be of the form of X-conditioned distributions over Y, indicated by $f(x \in X, y \in Y)$ (i.e., $P(y_f \mid f, q) = f(q, y_f)$). Any restrictions on f are imposed by P(f, h, d, c).

8) The "likelihood" is P(d | f). It says how d was generated from f. It is "vertical" if P(d | f) is independent of the values $f(x, y_f)$ for those $x \notin d_X$. As an example, the conventional iid likelihood

the learning curve - often an object of major interest - is the same for both errors over some region of interest.

viii) Second, although it's 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.

As an example, let s be the empirical misclassification rate between a hypothesis and the target over the training set, m the size of the training set, c'_{iid} the misclassification rate over all of the input space (the iid generalization error), and c'_{OTS} the misclassification rate over that part of the input space lying outside of the training set. Assume a uniform distribution over the input space, a uniform prior over target input-output relationships, and a noise-free iid likelihood. Then $P(s \mid c'_{iid}, m)$ is just $(c'_{iid})^{sm} (1 - c'_{iid})^{(m-sm)} C^{m}_{sm}$, 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} towards heads). On the other hand, $P(s \mid c'_{OTS}, m)$ is independent of c'_{OTS}. (This is proven in section 6 below.)

ix) Third, often it is more straight-forward 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 OTS error well-approximates iid error).

As an example, OTS error results presented below mean that when the training set is much smaller than the full input space, $P(c'_{iid} \mid s, m)$ is (arbitrarily close to) independent of s, if the prior over target input-output relationships is uniform. This holds despite VC results saying it is highly unlikely for c'_{iid} and s to differ significantly, no matter what the prior.

None of this means that the conventional error measure is "wrong". No claim is being made that one should not test with the same process that generated the training set, to avoid having error measure accuracy on the training set as well as off it. 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.

2. THE EXTENDED BAYESIAN FORMALISM

This paper uses the Extended Bayesian Formalism [Wolpert 1994a, Knill et al. 1994,Wolpert 1992]. In the current context, the EBF is just conventional probability theory, applied to the case where one has a dif-

grows. If one doesn't 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 errors is equivalent to requiring that test sets and training sets be distinct. (See [Wolpert 1994a].)

iii) In artificial intelligence - one of the primary fields concerned with supervised learning - the emphasis is often exclusively on generalizing to as yet unseen examples.

iv) 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) doesn't 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). So we will *only* be interested in OTS error.

v) 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's hard to see why one wouldn't want to distinguish them.

vi) 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 which 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.

vii) 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

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.

Viewed another way, this paper is an analysis of the "geometry" inherent in supervised learning before any particular probability distributions have been imposed. It is an analysis of supervised learning's skeleton, so to speak.

The work in this paper builds upon the analysis in [Wolpert 1992, 1993]. Some aspects of that analysis are nicely synopsized in [Schaffer 1993]. [Schaffer 1993] 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 1994] for related work in the field of combinatorial optimization.

1. 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, it is common practice to try to avoid fitting the training set exactly when one learns from that set, i.e., 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 et al. 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.

i) In the low-noise regime, optimal behavior on the training set is determined by look-up table memorization, and the only interesting issues concern OTS behavior.

ii) 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

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.

Next there is a discussion of one implication of these results: 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. It is then pointed out that this equivalence of expected errors between cross-validation and anti-cross-validation does not mean they have equivalent minimax properties. Indeed, it may be that cross-validation has better minimax properties than anti-cross-validation, and therefore can be *a priori* justified in that sense.

The analysis up to this point does not rule out the possibility that there are targets for which a particular learning strategy works well compared to another one. To address this issue, section 4 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 that, as a particular example, cross-validation can not be justified as a Bayesian procedure. I.e., there is no prior 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). Section 5 discusses 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 here that no algorithm is superior to its "randomized" version, and in this sense one can not *a priori* justify any particular learning algorithm, even for a quadratic loss function.

Section 6 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 can **not** say: if 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 "membership queries" algorithms and "punting" algorithms (those that may refuse to make a guess) are discussed.

Finally, section 7 presents some open issues.

It can not be emphasized enough that no claim whatsoever is being made in this paper that all algorithms

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: "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"; "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, Wolpert 1994ab].

Usually 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 paper presents a preliminary investigation of this issue. The primary tool used for this investigation is off-training set (OTS) generalization error, i.e., generalization error for test sets that contain no overlap with the training set. (In conventional generalization error such overlap is allowed.) Section 1 explains why such a measure of error is of interest. Those who already accept that OTS error is of interest can skip this section.

Section 2 presents the mathematical formalism used in this paper.

Section 3 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. As an example, it is shown that A is equivalent to B when B is an algorithm that chooses between two hypotheses

OFF-TRAINING SET ERROR AND *A PRIORI* DISTINCTIONS BETWEEN LEARNING ALGORITHMS

by

David H. Wolpert

The Santa Fe Institute, 1660 Old Pecos Trail, Suite A, Santa Fe, NM, 87501, dhw@santafe.edu

SFI TR 94-12-123

Abstract: This paper uses off-training set (OTS) error to investigate the assumption-free relationship between learning algorithms. 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 generalizer with largest cross-validation error). On the other hand, for loss functions other than zero-one (e.g., quadratic loss), there are *a priori* distinctions between algorithms. However even for such loss functions, any algorithm is equivalent on average to its "randomized" version, and in this still has no first principles justification in terms of average error. Nonetheless, it may be that (for example) cross-validation has better minimax properties than anti-cross-validation, even for zero-one loss. This paper also analyzes averages over hypotheses rather than targets. Such analyses hold for all possible priors. Accordingly they prove, as a particular example, that cross-validation can not be justified as a Bayesian procedure. In fact, for a very natural restriction of the class of learning algorithms, one should use anti-cross-validation rather than cross-validation (!). This paper ends with a discussion of the implications of these results for computational learning theory. It is shown that one can **not** say: if 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. 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.