Neural Network Interest Group

Título/*Title*:
No Free Lunch Theorems. An Introduction

Autor(es)/*Author(s)*:
J. P. Marques de Sá

Relatório Técnico/*Technical Report* No. 4 /2006

FEUP/INEB, Rua Dr. Roberto Frias, 4200-465, Porto, Portugal

Título/*Title*:

No Free Lunch Theorems. An Introduction

Autor(es)/*Author(s)*:

J.P. Marques de Sá

Relatório Técnico/*Technical Report* No. 4 /2006

Publicado por/*Published by*: NNIG. http://paginas.fe.up.pt/~nnig/

November 2006

Contents

1 Introduction

During a long time there was a widespread belief that there could exist, in general, a better learning algorithm than any other one, a better optimization algorithm than any other one, and so on and so forth.

> In the artificial neural network area how many times haven't we heard claims that SVM's are, in general, better than MLP's?

Since 1992 a whole series of theoretical results, pioneered by David Wolpert, shook down these beliefs.

These results are known as no-free-lunch theorems or NFLT for short. In rough terms they say that

Without prior knowledge of certain conditions satisfied by the problems they apply to all learning/search algorithms are equally good on average.

In other words, no particular algorithm provides a "free lunch" in terms of outperforming all other ones, **on average**.

2 No Free Lunch for Pattern Classification

2.1 The EBF Formalism

The Extended Bayesian Formalism (EBF) is a general framework that allows one to analyze properties of learning algorithms, encompassing other frameworks: Bayesian, statistical physics (simulated annealing), PAC and VC formalisms.

Henceforth random variables are denoted by capital letters; their instances by corresponding lower-case letters.

Consider (we follow Wolpert DH, 1995):

- An input and output spaces, X and Y.
- Training set: $d = \{(x_i, y_i)\}$, $i = 1,\ldots, n$. (Sometimes it may be convenient to stress the membership of the (x_i, y_i) pairs of d by writing: $(d_X(i), d_Y(i))$.)
- A specific (discrete or continuous) input-output relationship f: $P(y \mid x, f) = f_{x,y}$. A deterministic relationship corresponds to a delta function.
- H: the r.v. associated to a discrete set of hypothesis (set of parameters to be learned; e.g. $\{w_i\}$ in NN).
- $P(h \mid d)$: Probability of algorithm producing h when trained with d. For deterministic algorithms $P(h \mid d)$ is a delta function; for stochastic algorithms can be a broad distribution.
- C: the error (or cost or loss function) of the classifier. The *generalization error function* is the expectation value $E[C \mid h, f, d]$. For instance, the average misclassification rate error might be $E[C \mid h, f, d] = E[C \mid h, f] = \sum_x P(x)[1 - \delta(f(x), h(x))]$.

There are two aspects at the core of the NFLT for pattern classification:

1. The *Extended Bayesian Formalism* (EBF). This is just the usual Bayesian formalism (use of the prior $P(f)$ and likelihood $P(d \mid f)$ in order to compute the posterior $P(f \mid d)$) with the addition of $P(h \mid d)$. Thus, EBF is the probabilistic description of the quadruple $\{H, F, D, C\}$.
2. Use of the *misclassification rate error function*, *Er*, in two forms:

C is independent of D:

$$C = Er(F,H,D) = \sum_x P(x)\left[1 - \delta(F(x),H(x))\right] \qquad \text{("i.i.d. error")}$$

C is dependent on D:

$$C = Er(F,H,D) = \frac{\sum_{x \notin D_X} P(x)\left[1 - \delta(F(x),H(x))\right]}{\sum_{x \notin D_X} P(x)} \qquad \text{("}\textit{off-training-set}\text{ error")}$$

Reasons to evaluate off-training-set behavior of c:

i. The only real interesting issue is off-training-set behavior. Behavior in the training set is provided by simple table look-up.

ii. For either table look-up or test sets overlapping the training set, the upper bound on test set error shrinks as the training set grows.

iii. Very often the process generating the training set is not the same as that of test sets.

2.2 NFLT for Pattern Classification

Theorem 1. $E[C \mid d]$ can be written as a (non-Euclidian) inner product between the distributions $P(h \mid d)$ and $P(f \mid d)$:

$$E[C \mid d] = \sum_{h,f} Er(h,f,d)P(h \mid d)P(f \mid d)$$

The average generalization error given a certain d and taking into account all possible f and h depends on the "alignment" of $P(h \mid d)$ with $P(f \mid d)$. Unless one can prove that $P(f \mid d)$ has a certain form *one cannot prove how well* $P(h \mid d)$ *is aligned with* $P(f \mid d)$ *and therefore cannot prove anything concerning generalization performance.*

Let us now consider the expected **off-training-set error** of the k^{th} learning algorithm, for a given target function, f, and fixed training-set d:

$$E_k[C \mid f,d] = \sum_h \frac{\sum_{x \notin D} P(x)(1 - \delta(f(x),h(x)))}{\sum_{x \notin D} P(x)} P_k(h \mid d)$$

Similar definitions for other averages.

Theorem (NFLT for classification): For any two learning algorithms, $P_1(h \mid d)$ and $P_2(h \mid d)$, the following are true independent of the sampling distribution $P(x)$:

i. Uniformly averaged over all target functions f, $E_1[C \mid f, n] - E_2[C \mid f, n] = 0$.
ii. For any fixed training set d, uniformly averaged over all target functions f, $E_1[C \mid f, d] - E_2[C \mid f, d] = 0$.
iii. Uniformly averaged over all $P(f)$, $E_1[C \mid n] - E_2[C \mid n] = 0$.
iv. For any fixed training set d, uniformly averaged over all $P(f)$, $E_1[C \mid d] - E_2[C \mid d] = 0$.

This formulation of the NFLT is in Wolpert DH, 1995 and 2001. The proofs are in Wolpert DH, 1992 and Wolpert DH, 1996.

Interpretation:

1. $\sum_f \sum_{d,|d|=n} (E_1[C \mid f, n] - E_2[C \mid f, n]) P(d \mid f) = 0$; $P(f) = 1/|F|$

On average terms (for all f) and if all target functions are equally likely, there are no "good" algorithms, $P_1(h \mid d)$, outperforming "bad" ones, $P_2(h \mid d)$, i.e. with $E_1[C \mid f, n] < E_2[C \mid f, n]$.

2. $\sum_f (E_1[C \mid f, d] - E_2[C \mid f, d]) = 0$; $P(f) = 1/|F|$

Even if we fix d, then averaged over all target functions there is no algorithm yielding an off-training-set error lower that any other one.

3. $\sum_f \sum_{d,|d|=n} (E_1[C \mid f, n] - E_2[C \mid f, n]) P(d \mid f) P(f) = 0$

The same as 1 holds when taking into account $P(f)$ (i.e. for non-uniform target distributions).

$E_1[C \mid n] - E_2[C \mid n]$ is the average over all priors: $\sum_f (E_1[C \mid f, n] - E_2[C \mid f, n]) P(f)$

4. $\sum_f (E_1[C \mid f, d] - E_2[C \mid f, d]) P(d \mid f) P(f) = 0$

The same as 2 holds when taking into account $P(f)$ (i.e. for non-uniform target distributions).

Thus:

Without prior knowledge of $P(f \mid d)$ any algorithm performs on average as well as random guessing.

However, learning algorithms can differ in that:

1. For particular non-uniform $P(f)$, different algorithms have different data-conditioned risk: $E_k[C \mid f, d]P(d \mid f)P(f)$.
2. For some algorithms there is a distribution-conditioning quantity (e.g., f) for which that algorithm is optimal.
3. For some pairs of algorithms (A, B) the NFLTs are met by having many cases where A is slightly worse than B, and a few where A beats B by a lot.

2.3 Examples

1.
(Forster MR, 1999) provides the following simple illustrative example:
Consider an imaginary universe with 2 days and in each day one object which may be a sphere (S) or cube (C).
There are 4 possible histories: (S,S), (S,C), (C,S), (C,C).
We want a learning rule that predicts the object on the 2nd day based on the observation of the 1st day. There are 4 learning rules

Rule #1	Predict the same object
Rule #2	Predict a different object
Rule #3	always predict sphere
Rule #4	always predict cube

Consider first the uniformity of the histories: P(S,S) = P(S,C) = P (C,S) = P (C,C)= ¼
The probability of correct prediction is then the same for all rules: ½.

Now, consider "uniformity of nature": P(S,S) = P (C,C)= ½; P(S,C) = P (C,S) = 0.
The probabilities of correct prediction are: 1, 0, ½, ½ . For the rule #1 we have a perfect alignment of $P(h \mid d)$ with $P(f \mid d)$.

2.
Duda RO, Hart PE, Stork DG (2001) provide a simple Example and an illustration of the NFLT implications.

The following examples are from (Wolpert DH, 1995).

3.

- Deterministic learning algorithm trained with a particular d. Risk of interest: $E[C \mid d]$.
- **Empirical misclassification error** is s: $E[C \mid s, d] = E[C \mid d]$.
- s is low and VCD is low.

Will $E[C \mid d]$ be low (since VCD is low)? (Common interpretation of VC theorems.)

Theorem 2.iv provides no such assurance concerning off-training-set error.

4.

d is split into two parts, d_1 and d_2. Training is done in d_1, and d_2 is used as a "test set".

The error function is applied to both d_1 and d_2 but there is no consideration of off-training-set error. The NFLTs do not apply.

5.

The same as before but now we are interested in the *off-training-set Er*(f, h, d), where *off-training-set* means off all of d. The NFLTs apply: the behavior on d, including the behavior on d_2, tells us nothing about c, on average. (If this were not the case, behavior on d could be used to select the best algorithm.)

6.

The same as before but now we define off-training-set as all those pairs (x, y) which are not in d_1. Moreover, we take as d_2 the remaining pairs of d that cannot be found in d_1 (no overlap). *If our definition of off-training-set is no overlap*, we then take

$$E_k[C \mid f, d] = \sum_h \frac{\sum_{x \notin D} P(x)(1 - \delta(f(x), h(x)))}{\sum_{x \notin d_1} P(x)} P_k(h \mid d)$$

That is, our error function runs over d_2 and $X - d_X$. In this case the NFLTs do not apply. Indeed the behavior on d_2 tells us something about the likely c behavior (confidence interval).

7.

In certain situations the expected off-training-set error grows as the size of the training set increases, even if one uses the Bayes-optimal learning algorithm.

2.4 The Bayesian and VC Frameworks

The analysis of these and other two frameworks is given in (Wolpert DH, 1995).

2.4.1 The Bayesian Framework

In the light of the previous theorems and a specific theorem for this framework (Theorem 3 of Wolpert DH, 1995) the main conclusions are:

- Known distributions: $P(f)$ (prior), $P(d \mid f)$ (likelihood). Computed distribution: $P(f \mid d)$ (posterior; computed by Bayes' Theorem).

- This framework is concerned with $E(C \mid d)$. Given any particular form for $Er(f, h, d)$ we uniquely specify $E(C \mid d)$ for any particular algorithm $P(h \mid d)$. We are also able to find the $P(h \mid d)$ minimizing $E(C \mid d)$ (the Bayes-optimal generalizer).

- The Bayesian framework might appear to imply that in the absence of noise one should search for an h which agrees often with d. This is not the case.

- Even if $P(f)$ is larger for "simpler" f in many circumstances the optimal guess is not given by the simplest f consistent with the data.

2.4.2 The VC framework

- The VC framework is concerned with confidence intervals $P(|c - s| > \varepsilon \mid f, n)$ for variable ε.

- The VC dimension characterizes the support (over all h's) of $P(h \mid d)$. Many bounds on $P(|c - s| > \varepsilon \mid f, n)$ depend only on n, VCD and ε.

- In the illustrative single-h case, where $P(h \mid d)$ always guesses the same h, c does not vary in $P(|c - s| > \varepsilon \mid f, n)$; rather s does.

- In the coin-tossing analogy the VC framework's distribution of interest is analogous to the coin-flipping distribution $P(|c - s| > \varepsilon \mid c, n)$. However, our interest is on $P(|c - s| > \varepsilon \mid s, n)$.

- The behavior of $P(|c - s| > \varepsilon \mid s, n)$ for off-training-set error is not currently well understood.

3 No Free Lunch for Cross-Validation

An algorithm using cross-validation to choose amongst a pre-fixed set of algorithms is just a way of setting $P(h \mid d)$. By the preceding NFLTs it's no better on average than any other algorithm. However, since cross-validation can only be viewed as setting $P(h \mid d)$ when the set of generalizers is pre-fixed, NFLTs says nothing about cross-validation in general.

The following example (Zhu H, Rohwer R, 1996) is in fact discussed in (Wolpert, 1995).

Example of Zhu H, Rohwer R (1996):

A dataset $\{x_i\}$ with n elements generated by a Gaussian distribution $N(\mu,1)$. Three estimators of μ:
A: The optimal (ML, LSE) estimator: $\bar{x}$
B: A bad estimator: $x^* = \max\{x_i\}$
C: A cross-validation (CV) estimator: generate an extra element x_{n+1}; if $\|\bar{x} - x_{n+1}\| < \|x^* - x_{n+1}\|$ take estimate A, otherwise take B. ($\|\ \|$ is the square norm.)

The mean square errors we have obtained for a run with 10^5 samples and $n = 16$:

A: 0.0625 B: 3.4037 C: 0.5695
The value of A reproduces the theoretical mean squared error: 1/16 = 0.0625.

The CV estimate is rather bad. Close analysis reveals that rarely B is preferred over A; but when it is preferred it is often for the wrong reason. For instance (Zhu H, Rohwer R, 1996):
"In 10000 samples about 30 are such that B is better than A. However, based on one extra point, C will prefer B for about 2000 cases. Among them, only about 15 samples are for the right reason, that B is genuinely better."

Example of Goutte C (1997):

A re-analysis of the previous example, including a stricter CV estimator:

C_{LOO}: A leave-one-out version of estimator C: consider the same n-sized dataset and calculate average distances (square norm) between an estimate (A or B) evaluated at n-1 points and the remaining point. Choose the estimate with smaller distance.

The mean square errors we obtained for a run with 10^5 samples and $n = 16$:

A: 0.0625 B: 3.4037 C: 0.5695 C_{LOO}: 0.0625

Let $\bar{x}_k$ denote the mean of the points *excluding* x_k:

$$\bar{x}_k = \frac{n\bar{x} - x_k}{n-1} \quad \Rightarrow \quad \bar{x}_k - x_k = \frac{n}{n-1}(\bar{x} - x_k)\,.$$

For the A estimator C_{LOO} computes: $CV_A = \left(\frac{n}{n-1}\right)^2 \frac{1}{n}\sum_{k=1}^{n}(\bar{x} - x_k)^2$

For the B estimator one has to compute the maximum on $n-1$ points. Let x_k^* denote the maximum of the points *excluding* x_k: $x_k^* = \max\{x_i \setminus x_k\}$. Now, x_k^* is equal to $x^* = \max\{x_i\}$ in $n-1$ subsets and equal to $x^{**} = \max\{x_i \setminus x^*\}$ in the remaining subset (say, the n^{th} subset). Therefore, for the B estimator C_{LOO} computes:

$$CV_B = \sum_{k=1}^{n-1}(x^* - x_k)^2 + \frac{1}{n}(x^{**} - x^*)^2$$

Using Huygens' formula: $CV_B = \left(\frac{n-1}{n}\right)^2 CV_A + \left(\bar{x} - x^*\right)^2 + \frac{1}{n+1}(x^* - x^{**})^2$

In order for CV_B to be smaller than CV_A, x^* should have to be close to both $\bar{x}$ and x^{**}, a very rare occurrence.

Using 2-fold cross-validation instead of LOO will produce estimates worse than A (e.g., $C_{CV2} = 0.0628$).

"We now know that there is "no free lunch" for cross-validation. However, the task of exhibiting an easily understandable, non-degenerate case where it fails has yet to be completed."

Further Insight on Cross-Validation (Rivals I, Personnaz L, 1999)

By analyzing the IMSE of LOO in the estimation of linear model parameter vectors, Rivals & Personnaz show that LOO estimates are biased, whereas statistical tests are based on unbiased estimates (when certain assumptions are met). As a consequence, *although LOO is not subject to NFL criticism, it often performs rather poorly in model selection, worse than classic statistical tests* (whenever these can be applied).

4 No Free Lunch for Early Stopping

The (only?) work of reference is Cataltepe Z *et al.* (1999).

Let the training set be denoted as $D = \{(\mathbf{x}_i, f_i)\}$, $i = 1,\ldots, n$; $f_i = f(\mathbf{x}_i)$
The model fitting the training data is: $g_{\mathbf{v}}(\mathbf{x}_i)$ with parameter vector $\mathbf{v}$.
The quadratic training error is:

$$E_T(\mathbf{v}) = \frac{1}{n}\sum_{i=1}^{n}(g_{\mathbf{v}}(\mathbf{x}_i) - f_i)^2$$

The generalization error of model **v** is:

$$E(\mathbf{v}) = E_{\mathbf{x}}[(g_{\mathbf{v}}(\mathbf{x}) - f(\mathbf{x}))^2]$$

Now, consider $\mathbf{v}_T$ to be the model corresponding to a local minimum of the training error E_T.
Assume stopping at $E_\delta = E_T(\mathbf{v}_T) + \delta$, $\delta \geq 0$ and the *early stopping set* of models:

$$\mathbf{W}_\delta = \{\Delta\mathbf{v} : E_T(\mathbf{v}_T + \Delta\mathbf{v}) = E_\delta\}$$

The mean generalization error at training set error level E_δ is:

$$E_{mean}(E_\delta) = \int_{\Delta\mathbf{v}\in\mathbf{W}_\delta} P_{\mathbf{W}_\delta}(\Delta\mathbf{v}) E(\mathbf{v}_T + \Delta\mathbf{v}) d\Delta\mathbf{v}$$

Linear models ($g_{\mathbf{v}}(\mathbf{x}) = \sum v_i \phi_i(\mathbf{x})$):

Lemma 1. When all (linear) models with training error E_δ are equally likely to be chosen as the early stopping solution, the mean generalization error is at least as much as the generalization error of the training error minimum:

$$E_{mean}(E_\delta) = E(\mathbf{v}_T) + \beta(\delta), \quad \beta(\delta) \geq 0$$

Theorem 1.When all models with the same training error are equally likely to be chosen as the early stopping solution, the mean generalization error is an increasing function of the early stopping training error:

$$\text{For } 0 < \delta_1 < \delta_2, \quad E_{mean}(E_{\delta_1}) < E_{mean}(E_{\delta_2}).$$

Non-linear models:

Similar results hold when considering an adequate location of the training error minimum and a neighborhood from it. Let **v*** denote a minimum of the generalization error.

Theorem 2. Assume:

$$\mathbf{v}_T - \mathbf{v}^* = O(1/\sqrt{n}),\ \delta \geq 0,\ \delta = O(1/n), \text{ and } E_\delta = E_T(\mathbf{v}_T) + \delta + O(1/n^{1.5})$$

When all models with training error E_δ are equally likely to be chosen as the early stopping solution, their mean generalization error is:

$$E_{mean}(E_\delta) = E(\mathbf{v}_T) + \beta(\delta) + O(1/n^{1.5}), \quad \beta(\delta) \geq 0$$

Early stopping as a regularization method:

When performing early stopping based on either weight-decay or validation set minimum error, *models with small weights are favored*; i.e., $P_{\mathbf{W}_\delta}(\Delta\mathbf{v})$ is *not* uniform. Thus, early stopping may be beneficial (if $P_{\mathbf{W}_\delta}(\Delta\mathbf{v})$ agrees with the probability of the target function).

Cataltepe Z *et al.* (1999) performed experiments for linear models showing that early stopping with weight decay is beneficial.

5 No Free Lunch for Search

5.1 Optimization Problems

Let us consider the optimization problem: searching an extreme of a cost function *f*.

$$f: X \to Y \qquad (F \equiv \{f\})$$

How *search algorithms* work:

- Given *m* points $(x, y)^m \in (X \times Y)^m$ extrapolate to a new, hopefully better (lower/higher) cost point, $x' \in X$: a *search strategy*.
- Extrapolation can either be deterministic (e.g., branch-and-bound) or stochastic (e.g., genetic algorithm).

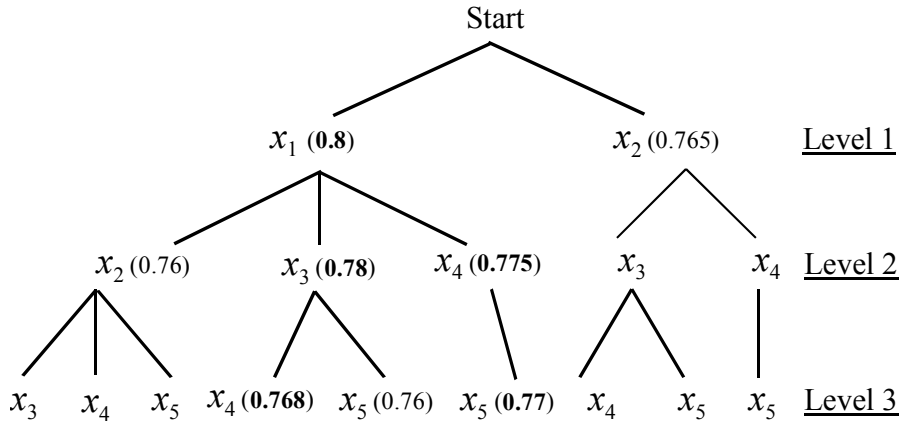
Example
Feature selection with a monotonic criterion, *J* (e.g. Anova distance). We want the highest possible *J* for an "optimal" feature subset *B* of a feature set *D*.

$$X = \mathcal{P}(D); \qquad Y = \Re_J;$$

$$\text{Cost function:} \quad f: \mathcal{P}(D) \to \Re_J$$
$$B \to J(B)$$

■

Example of branch-and-bound
Search an optimal 2-feature subset, *B*, with *J*(*B*), out of a 5-feature set $D = \{x_1, x_2, x_3, x_4, x_5\}$.



Strategy

1. Take $m = 1$; $B = D$;
2. Repeat until $m = 3$ (branch)
   Find out which subset $B - \{x_i\}$, with $5-m$ features has the best (highest) criterion. $B \leftarrow \text{opt}(B - \{x_i\})$.
   $J_{curr} = J(\text{opt}(B - \{x_i\}))$ is the *current bound*.
3. While ($\exists$ non-followed paths with $J < J_{curr}$)
   Backtrack.

■

Example of genetic algorithm

A feature subset is a string of 0s and 1s (selected features): chromosome B.
Each chromosome has a fitness $J(B)$.

Strategy

1. Select an initial population $G = \{B\}$ with m chromosomes.
2. While stopping criterion not verified (e.g. there is no B in G with $J(B) > \Delta$)
   - Select chromosomes for reproduction with probability $\propto J(B)$.
   - Apply crossover (e.g. swap parts of parent strings) with probability P_X.
   - Apply mutation (e.g. invert a gene) with probability P_M.
   - Update G with the offspring.

■

Note that *exhaustive search* is *not* a search strategy. It's table lookup.

5.2 NFLT for Search

Let us now analyze the search process in general.
(We follow Wolpert DH, Macready WG, 1995)

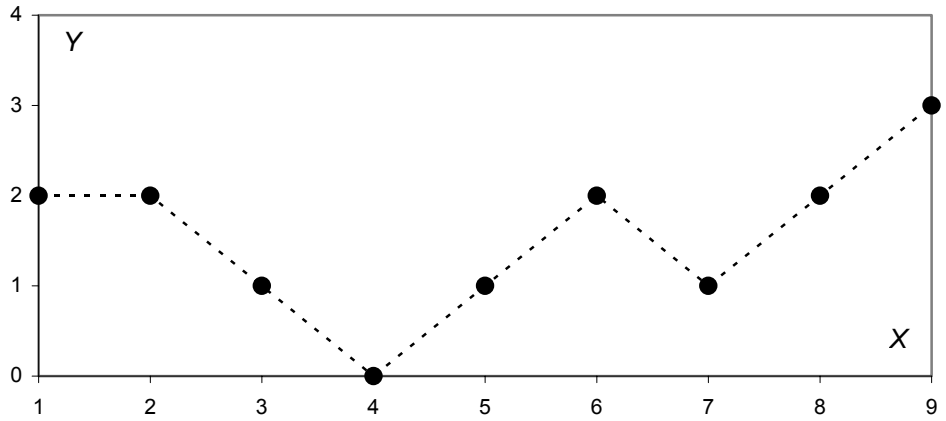
- For simplicity take X and Y to be finite.
- Define $d_m \equiv \{x(i), y(i)\} \in D_m, \quad i = 1,...,m$ a set of m distinct search points and associated costs. Often the ordering is according to the time the points are visited.
- Let $\vec{c}$ represent the histogram of cost values that an algorithm, a, obtains on a particular cost function f. Note that $\vec{c}$ is a vector of length $|Y|$ and provides a measure of algorithmic performance.

Example

$X = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$
$Y = \{0, 1, 2, 3\}$

Imagine we had to search the minimum of:

x_i	1	2	3	4	5	6	7	8	9
$y_i=f(x_i)$	2	2	1	0	1	2	1	2	3



There are $|X|^{|Y|} = 9^4 = 6561$ such functions.

Consider the deterministic algorithms:

a_1 ("go forward"): start at x_1 and progress to x_{i+1} until finding the first local minimum.

a_2 ("go backward"): start at x_9 and progress to x_{i-1} until finding the first local minimum.

a_3 ("binary search"): start at $x_{low} = x_1$ and $x_{up} = x_9$; at each step progress to the middle of $[x_{low}, x_{up}]$ and select the half interval with lower costs for the next step.

We then have for $m = 6$ the following d_6 paths:

a_1	
x_i	y_i
1	2
2	2
3	1
4	0
5	1
4	0

a_2	
x_i	y_i
9	3
8	2
7	1
6	2
7	1
6	1

a_3		
x_i	y_i	Next interval
1	2	
9	3	[1,9]
5	1	[1,5]
3	1	[3,5]
4	0	[3,4]
4	0	

Now consider the stochastic algorithms:

a_4 ("random choice with replacement"): at each step make a random choice of which point to visits. At the end the minimum value with non null visits is selected.

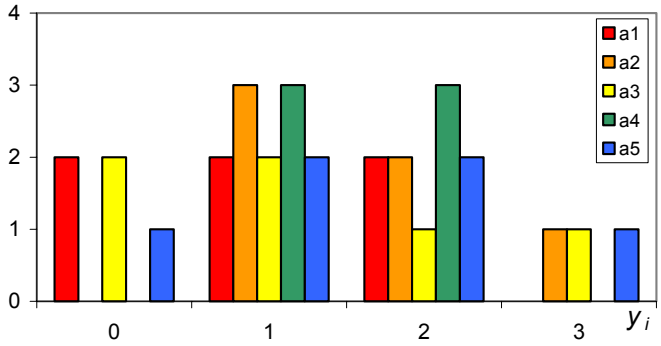
a_5 ("random choice without replacement"): as before, with the only difference that after step 2 each point with higher cost is removed.

For $m = 6$ the following d_6 paths are possible outcomes:

a_4	
x_i	y_i
7	1
5	1
1	2
8	2
5	1
6	2

a_5		
x_i	y_i	Points removed
2	2	
7	1	2
5	1	
9	3	9
2	2	2
4	0	

The histograms are:



■

We now consider the probability that a histogram $\vec{c}$ will be obtained under m distinct cost evaluations of algorithm a on f: $P(\vec{c} \mid f, m, a)$. All algorithms use the same amount of information: m cost evaluations (search points).

Theorem (NFLT for search): For any pair of algorithms, a_1 and a_2,

$$\sum_f P(\vec{c} \mid f, m, a_1) = \sum_f P(\vec{c} \mid f, m, a_2). \quad (1)$$

$P(\vec{c} \mid f, m, a)$ is independent of a when we average over all cost functions.

The proof in (Wolpert DH, Macready WG, 1995) is first given for deterministic algorithms without backtrack. Later the proof is extended for algorithms with backtrack or of stochastic search.

Consequences of this Theorem and other related Theorems:

- The expected histogram, and therefore the expected performance, $E[\vec{c} \mid f, m, a] = \sum_{\vec{c}} \vec{c} P(\vec{c} \mid f, m, a)$, is on average the same for all algorithms.

- After m cost evaluations of an algorithm, if a_1 has better performance over a_2 in some subset of F, a_2 must outperform a_1 in the remaining subset.

  "There can be no search algorithm that outperforms all others on all problems."

- Note that:

  $$P(\vec{c} \mid m, a) = \sum_f P(\vec{c} \mid f, m, a) P(f \mid m, a) = \sum_f P(\vec{c} \mid f, m, a) P(f) \quad (2)$$

  since f is independent of m and a.

  If we know nothing about f, all f, are equally likely: $P(f) = 1/|Y|^{|X|}$. Then, by the Theorem:

  $$P(\vec{c} \mid m, a) = (1/|Y|^{|X|}) \sum_f P(\vec{c} \mid f, m, a)$$

  is independent of a.

- Equation (2) can be seen as a dot product in F-space of $\vec{v}_{c,a,m} = P(\vec{c} \mid f, m, a)$ and $\vec{p} = P(f)$. Any knowledge on the cost function goes into the prior $\vec{p}$. Given a certain c and m the optimal algorithm is the one with largest projection onto $\vec{p}$.

- Let us consider a meta-search algorithm that by observing how well two competing search algorithms have done so far (in m steps) for a fixed cost function f decides which algorithm to use further. The NFL Theorem for this scenario tells us that on average such meta-search algorithms behave equally: observing how well an algorithm has done so far tells us nothing about how well it will do if we continue to use it in the same f.

5.3 A Simple Explanation

Ho YC and Pepyne DL (2002) provide a "Simple Explanation" of the NFLT for search, based on the properties of *fundamental matrices*, *F-matrices*, such as:

	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7
x_0	y_0	y_1	y_0	y_1	y_0	y_1	y_0	y_1
x_1	y_0	y_0	y_1	y_1	y_0	y_0	y_1	y_1
x_2	y_0	y_0	y_0	y_0	y_1	y_1	y_1	y_1

for $|X| = 3$ and $|Y| = 2$.

- For a fundamental matrix the row sums (averages) are equal.
- The sub-matrix obtained by eliminating row i and all columns j such that $F_{ij} \neq y$ is an F-matrix.

For instance, the sub-matrix obtained by eliminating row 0 and all columns such that $f_j(x_0) \neq y_0$:

	f_0	f_2	f_4	f_6
x_1	y_0	y_1	y_0	y_1
x_2	y_0	y_0	y_1	y_1

The NFLT follows by considering the F-matrix whose rows are strategies, whose columns are all possible optimization problems and whose $f(x)$ are performance of strategy x on problem f (after m steps).

	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7
a_0	c_0	c_1	c_0	c_1	c_0	c_1	c_0	c_1
a_1	c_0	c_0	c_1	c_1	c_0	c_0	c_1	c_1
…	…	…	…	…	…	…	…	…
a_n	c_0	c_0	c_0	c_0	c_1	c_1	c_1	c_1

c_0 could mean "bad" and c_1 "good".

- Averaged over all problems (columns of F) all strategies (row averages) are equal.

5.4 Further Results

Whereas Wolpert and Macready (1997) demonstrate NFL for the class of all functions, Schumacher et al. (2001) show that NFL still holds iff the class of functions F is closed

under permutation (a permutation of a function simply permutes the y values for the same x values).

The NFL does not hold for:

- For some classes of extremely simple functions (Droste, Jansen and Wegener, 1999).
- When the class of functions concern functions having less than the maximum number of local minima or functions having less than the maximal possible steepness (Igel C, Toussaint M, 2001).
- For sets with bounded description length.
- $P(f)$ is a decreasing function of f's description length.

6 NFL Abuse

Besides erroneous interpretations of NFL results, one should be aware of more serious non-innocent abuses. See, for instance, (Shallit J, 2002).

References

I list below the references that I found particularly useful. An extensive list can be found in: No Free Lunch Theorems: http//www.no-free-lunch.org/.

Cataltepe Z, Abu-Mostafa YS, Magdon-Ismail M (1999) No Free Lunch for Early Stopping. Neural Computation 11, pp. 995-1009.

Droste S, Jansen T, Wegener I (1998) Perhaps not a Free Lunch but at Least a Free Appetizer. Technical Report ISSN 1433-3325, Dept. Computer Science, University of Dortmund

Duda RO, Hart PE, Stork DG (2001) Pattern Classification. John Wiley & Sons, Inc.

Forster MR (1999) Notice: No Free Lunches for Anyone, Bayesian Included. Phylosophy Dept., University of Wisconsin.

Goutte C (1997) Note on Free Lunches and Cross-Validation. Neural Computation, vol.9,No.6, pp. 1245-1249.

Ho YC, Pepyne DL (2002) Simple Explanation of the No-Free-Lunch Theorem and Its Implications. J. Optimization Theory and Applications, vol. 115, No. 3, pp. 549-570.

Igel C, Toussaint M (2003) On Classes of Functions for which No Free Lunch Results Hold. Information Processing Letter 86(6), pp. 317-321.

Rivals I, Personnaz L (1999) On Cross Validation for Model Selection. Neural Computation 11, pp. 863-870.
Schaffer C (1993) Overfitting Avoidance as Bias. Machine Learning, 10, pp. 153-178.
Schumacher C, Vose MD, Whitley LD (2001) The No Free Lunch and Problem Description Length. In Bayer H et al. eds., Gecco 2001 Proc.
Shallit J (2002) Book Review of "William Dembski. No Free Lunch: Why Specified Complexity Cannot be Purchased Without Intelligence". BioSystems, 66, pp. 93-99.
Streeter MJ (2003) Two Broad Classes of Functions for Which a No Free Lunch Result Does not Hold. Gecco 2003 Proc.
Wolpert DH (1992) On the Connection between In-sample Testing and Generalization Error. Complex Systems, 6, pp. 47-94.
Wolpert DH (1995) The Relationship Between PAC, the Statistical Physics Framework, the Bayesian Framework, and the VC Framework. In: The Mathematics of Generalization, Proc. SFI/CNLS Workshop (ed. Wolpert DH). Addison_wesley Pub. Co., pp.117-214.
Wolpert DH (1996) The Lack of a Priori Distinctions Between Learning Algorithms and the Existence of a Priori Distinctions Between Learning Algorithms. Neural Computation, 8:1341-1390, 1391-1421.
Wolpert DH (2001) The Supervised Learning No-Free-Lunch Theorems. NASA Ames Research Center. June 22, 2001.
Wolpert DH (2001) Any Two Learning Algorithms Are (Almost) Exactly Identical. NASA Ames Research Center. July 11, 2001.
Wolpert DH, Macready WG (1995) No Free Lunch Theorems for Search. Technical Report SFI-TR-95-02-010, Santa Fe Institute, Santa Fe, New Mexico.
Wolpert DH, Macready WG (1997) No Free Lunch Theorems for Optimization. IEEE Tr. Evolutionary Computing, 1:1, pp. 67-82.
Zhu H, Rohwer R (1996) No Free Lunch for Cross-Validation. Neural Computation 8, pp. 1421-1426.