A NO FREE LUNCH RESULT FOR OPTIMIZATION AND ITS IMPLICATIONS

A Thesis

Submitted to the McAnulty College

and Graduate School of Liberal Arts

Duquesne University

In partial fulfillment of the requirements for

the degree of Master of Science in Computational Mathematics

By

Marisa B. Smith

August 2009

A NO FREE LUNCH RESULT FOR OPTIMIZATION

AND ITS IMPLICATIONS

By

Marisa B. Smith

Approved May 5, 2009

APPROVED_____
Jeffrey Jackson, Ph.D., Professor of Computer Science
Department Chair
Department of Mathematics & Computer Science

APPROVED_____
John Kern, Ph.D., Associate Professor of Statistics
Department of Mathematics & Computer Science

APPROVED_____
Mark Mazur, Ph.D., Associate Professor of Mathematics
Graduate Director of Computational Mathematics
Department of Mathematics & Computer Science

APPROVED_____
Ralph L. Pearson, Ph.D.
Provost and Academic Vice President

ABSTRACT

A NO FREE LUNCH RESULT FOR OPTIMIZATION

AND ITS IMPLICATIONS

By

Marisa B. Smith

August 2009

Thesis Supervised by Dr. Jeffrey Jackson

The No Free Lunch (NFL) theorems for optimization tell us that when averaged over all possible optimization problems the performance of any two optimization algorithms is statistically identical. This seems to imply that there are no "general-purpose" optimization algorithms. That is, the NFL theorems show that, mathematically, any superior performance of an optimization algorithm on one set of problems is offset by inferior performance of that algorithm on the set of all other problems. In this thesis we consider the seemingly negative implications of the NFL theorems. We first extend a previous NFL theorem to get a new NFL result. We then use ideas from probability theory and cryptography to show that if we believe that extraordinarily small probability events will not happen, then there exists (at least) one algorithm that is indeed a general-purpose algorithm. Thus, the implications of the new NFL result are not as negative as expected.

ACKNOWLEDGMENT

First of all, I would like to thank my thesis advisor, Dr. Jeffrey Jackson, without whom I never would have been able to complete this work. The countless hours of discussion we had made what is presented here possible, and I greatly appreciate all of the time he was able to dedicate in helping me to complete this thesis. I would also like to thank Dr. John Kern and Dr. Mark Mazur for being readers. All of their feedback was extremely helpful in allowing me to reach my final goal. Finally, I would like to thank my family for all of their support and patience, and for understanding when I could not spend as much time with them as I would have liked because I had to do my "homework."

Table of Contents

Chapter 1

Introduction

The purpose of optimization is to find the "best" solution to a particular problem. One commonly studied branch of optimization is combinatorial optimization. In combinatorial optimization one deals with functions (problems) in which a finite search space $\mathcal{X}$ maps to a finite space of cost values $\mathcal{Y}$. That is, a function f is defined as $f : \mathcal{X} \mapsto \mathcal{Y}$, and the goal is to maximize (or minimize) $f(x)$ for $x \in \mathcal{X}$ [1]. An optimization algorithm is then some method of choosing x's in $\mathcal{X}$ in order to meet this goal. That is, it is some method of choosing x's such that $f(x)$ is as large (or small) as possible for a given f.

For this paper, the performance of optimization algorithms is studied. In particular, we are interested in whether there exist any "general-purpose" optimization algorithms. That is, do there exist any algorithms that never do worse than random guessing, and will do better on certain problems? The "no free lunch" (NFL) theorems seem to imply that the answer is no. These theorems tell us that, mathematically, if we average performance of any optimization algorithm over all possible functions, its performance is the same as every other optimization algorithm. So instances in which an algorithm does better than random are necessarily offset by instances in which it does worse than random (so, on average,

performance is the same as random guessing). From this, Wolpert and Macready infer that there are no general-purpose optimization algorithms [2]. In this paper, we give reason to question this inference.

1.1 Background on No Free Lunch Theorems

As mentioned previously, the "no free lunch" (NFL) theorems for optimization are a collection of theorems that say, roughly speaking, that when averaged over all possible problems, the performance of any pair of optimization algorithms is statistically equivalent [2]. In other words, if one algorithm performs better than another algorithm on one set of problems, this superior performance is necessarily offset by inferior performance on the set of all other problems. Because we are averaging over *all possible* functions, on average, the performance of the algorithms is the same. That is, there is "no free lunch."

The NFL theorems state that the average performance of "any pair" of algorithms is identical, and since any pair of algorithms could include one that randomly chooses the next point to explore in its optimization process, an alternative way to state the NFL theorems is to say that, on average, the performance of any optimization algorithm is no better (or worse) than random guessing. Thus, according to the NFL theorems, when trying to find the maximum of a function, random guessing will perform just as well on average as hill-climbing, an algorithm which at each step examines a set of neighboring points and moves to the one having the highest cost value (and perhaps if it becomes stuck in a local maximum, randomly chooses a new point [3]).

Therefore, contrary to what one might expect, an algorithm such as hill-climbing, which makes informed decisions at each step, performs, on average, no better than an algorithm that randomly chooses its next point. In fact, perhaps

even more counterintuitively, the NFL theorems tell us that, on average, hill-descending, which at each step moves to the neighboring point with the *lowest* cost value, performs just as well as hill-climbing when trying to find the maximum. Although this runs counter to the fact that, in practice, hill-climbing is widely preferred to hill-descending or random guessing if trying to find the maximum, in a strict mathematical sense they are equivalent. The reason for this, as mentioned earlier, is that the averaging is computed over *all possible* functions.

1.1.1 Some Intuition on Why the NFL Theorems Hold

To get a better understanding of why this is true, consider the following. An optimization algorithm takes some sequence of steps to find the maximum of a function. In particular, at each step an optimization algorithm has some fixed method of choosing an x given a history of (x, y) pairs. However, because we are considering *all possible* functions, the next x chosen could have any of the possible y-values, regardless of how x is chosen (i.e., regardless of the algorithm). So although for one particular function an algorithm might see a sequence of (x, y) pairs that leads to "good values" (larger y-values), this same sequence is present in some other function that will lead to "bad values" (smaller y-values). In fact, there are other functions that will lead to all values in between. That is, after seeing some initial sequence of (x, y) pairs, the next steps that an algorithm takes could lead anywhere if all possible functions are considered. Since this is true regardless of the algorithm, it is true of all algorithms. Thus, when one is considering all functions, algorithm performance is the same, on average.

An alternative point of view is to picture each function as having a particular terrain. While running an optimization algorithm on a particular function, each step reveals more and more of the terrain. However, seeing part of the terrain tells one nothing about the rest of the terrain if all terrains are possible (i.e., if all functions

are possible). For example, the part of the terrain one has seen so far might be very smooth and level, but the rest of the terrain is just as likely to be completely random as it is to continue to be smooth and level when one is considering all possible functions. Thus, averaged across all possible terrains (functions), algorithm performance is the same.

These examples of why the NFL theorems hold are merely to provide some intuition. Section 3.2.2 contains a formal NFL theorem and proof.

1.1.2 A Particular NFL Theorem of Interest

There are a number of NFL theorems for optimization, each pertaining to different algorithm types (e.g., deterministic or stochastic) and performance measures. One theorem of particular interest for this paper involves what are called choosing procedures. A choosing procedure can be thought of as a meta-algorithm that examines the performance of two optimization algorithms after running each a certain number of steps on an optimization problem, and based on these runs, chooses which of the two algorithms will be used for the remainder of the problem. Somewhat more formally, the scenario for this NFL theorem is as follows: (1) two optimization algorithms, a and a', are each run m steps on a particular problem; and (2) based on the performance of the two algorithms on these first m steps, a choosing procedure selects which of the two algorithms to use to continue the search (on unseen data). For this theorem, rather than averaging performance across all problems and saying that all algorithms are equivalent, we are instead averaging across all pairs of algorithms and saying that all choosing procedures are equivalent. Regardless, there is still "no free lunch," that is, mathematically, any increased performance of a choosing procedure on one pair of algorithms is offset by reduced performance on the remaining algorithm pairs. Similar to the theorems that seem to imply that there are no general-purpose optimization algorithms, this theorem

seems to imply that there are no general-purpose choosing procedures.

1.2 Objective of the Present Study

The first result of this thesis is the proof of an NFL theorem that is an extension of the choosing procedure situation described above. In particular, in the original choosing procedure NFL theorem, a pair of algorithms are run a given number of steps and based on performance on these initial steps a choosing procedure selects an algorithm. For our new NFL result, a pair of algorithms are run *multiple* times, each time with a new common starting x-value (we will call each run a training run), and then an algorithm is selected. Also, for the original theorem, once an algorithm is selected, performance is measured based on a continuation of the algorithm run. For the new theorem, once an algorithm is selected, performance is measured on a new algorithm run, starting from a new, previously unseen, starting value (we will call this the test run). Using techniques from the proof in [2] of the original choosing procedure situation (where each algorithm is run once), it will be shown that for this new setting, where a set of training runs are performed, any two choosing procedures are (still) mathematically equivalent. That is, when averaged over all possible pairs of optimization algorithms any two choosing procedures are equivalent in terms of the performance of their chosen algorithm on test runs.

The second result of this thesis will use a fundamental assumption of modern cryptography to show that despite the new NFL result, there exists (at least) one choosing procedure that is indeed better than another, in particular, better than random guessing. More specifically, we will show that for this "better" choosing procedure, when one of the optimization algorithms consistently outperforms the other on sufficiently many training runs, then it is extremely unlikely that the procedure will be "fooled" into selecting this algorithm when it is actually worse

than the other algorithm. That is, it is extremely unlikely that one algorithm will always perform better on many training runs but will more often than not perform *worse* on a new algorithm run. We will then show that with a reasonable number of training runs, the probability that the choosing procedure is fooled in such a way can be made so small that it is rational to always trust the choosing procedure. This then leads to our claim that our choosing procedure is better than a choosing procedure that randomly selects an algorithm. This is because the choosing procedure will not be fooled into making bad decisions, thus good performance will not be necessarily offset by bad performance. This then means that overall the procedure's average performance is better than random and hence it is a general-purpose choosing procedure.

Finally, this thesis discusses the implications of the results and how it is possible for both to hold. In particular, the results will be related to the well-known probability and decision theory problem, the St. Petersburg paradox, which similarly leads to two conflicting points of view. Also, relationships to current practice in the field of cryptography will be discussed. In doing so, it will be shown that although in a strict mathematical sense our new NFL theorem holds, the real-world implications are not as negative as expected.

Chapter 2

NFL Theorems for Optimization

In order to get to our new NFL result, it is first necessary to discuss the original NFL theorem from which it stems. Before doing this, we need to introduce the necessary definitions and notation that will be used.

2.1 Preliminaries

The NFL theorems are concerned with combinatorial optimization. Thus, the search space $\mathcal{X}$ and the space of possible cost values $\mathcal{Y}$ are finite (although perhaps quite large). An optimization problem f is a mapping from $\mathcal{X}$ to $\mathcal{Y}$, $f : \mathcal{X} \mapsto \mathcal{Y}$, and the space of all possible problems is $\mathcal{F} = \mathcal{Y}^{\mathcal{X}}$.

In establishing the NFL theorems, an oracle-based model of computation is used. In this view, during the optimization process, a particular $x \in \mathcal{X}$ is given to an "oracle" that then returns $f(x) = y \in \mathcal{Y}$ for the given optimization problem $f \in \mathcal{F}$. Performance is then evaluated based on the results produced from a certain number of function evaluations (calls to the oracle).

When performing an optimization problem, the set of function evaluations is called the sample. If m function evaluations are performed, then the sample is denoted $d_m \equiv \{(d_m^x(1), d_m^y(1)), \ldots, (d_m^x(m), d_m^y(m))\}$, where $d_m^x(i)$ is the ith

(time-ordered) search point in the sample and $d_m^y(i)$ is its corresponding cost value. Also, $d_m^x \equiv \{d_m^x(1), \ldots, d_m^x(m)\}$ is the ordered set of m points in the search space that have been explored, and $d_m^y \equiv \{d_m^y(1), \ldots, d_m^y(m)\}$ is the associated set of cost values. An important note here is that only distinct function evaluations are counted in these sets. That is, revisits are not counted as evaluations. This is because revisits complicate algorithm comparisons and can be easily avoided [2].

An optimization algorithm is a mapping from a previously visited (ordered) set of points to a single new (previously unvisited) point in $\mathcal{X}$. More formally, $a : d \in \mathcal{D} \mapsto \{x | x \notin d^x\}$, where $\mathcal{D} \equiv \cup_{m \geq 0} \mathcal{D}_m$ is the set of all samples of arbitrary size and $\mathcal{D}_m = (\mathcal{X} \times \mathcal{Y})^m$ is the space of all samples of size m. Note that by the definition above, an optimization algorithm is deterministic (although Wolpert and Macready show in [2] that NFL results apply to stochastic algorithms as well).

As previously mentioned, we are using an oracle-based model of computation. Thus, algorithm performance is measured by some function of d_m^y, the set of cost values after m iterations of the algorithm. A performance measure will be indicated by $\Phi(d_m^y)$ to reflect this relationship.

As an example, when trying to find the maximum of a function f, an appropriate performance measure might be the largest cost value (y-value) in d_m^y, $\Phi(d_m^y) : max_i\{d_m^y(i) : i = 1, 2, \ldots, m\}$. Another (perhaps better) way to evaluate algorithm performance that considers all elements in d_m^y, and does not "reward" an algorithm that happens to stumble upon one lucky guess, is to use the histogram of cost values. This histogram, denoted $\vec{c}$, is defined as follows:

$$\vec{c} = (c_{\mathcal{Y}_1}, c_{\mathcal{Y}_2}, \ldots, c_{\mathcal{Y}_{|\mathcal{Y}|}})$$

where $c_{\mathcal{Y}_i}$ is the number of times the cost value $\mathcal{Y}_i$ occurs in d_m^y. Note that this histogram (or vector) has one component for each possible $y \in \mathcal{Y}$, thus there are a

total of $|\mathcal{Y}|$ components. We can then apply some function that maps $\vec{c}$ to a "goodness" measure. That is, we apply some function that takes the histogram of cost values $\vec{c}$ and outputs some sort of ranking of how good this histogram is. For example, we could use $\Phi(\vec{c}) : \vec{c} \mapsto \mathbb{R}$ where larger values in $\mathbb{R}$ indicate a better ranking. We can then use this ranking as our performance measure. Note that this is just one possibility for $\Phi(\vec{c})$.

2.2 Choosing Procedure NFL Theorem

2.2.1 Definitions

The above definitions and notation are used to develop a framework for all of the NFL theorems presented in [2]. However, a few additional definitions are necessary that are specific to the choosing procedure NFL theorem introduced earlier. An obvious addition is the need for the definition of a choosing procedure. From [2]:

Definition 1 (Choosing Procedure - Wolpert and Macready)**.** *A* choosing procedure *is a rule that examines the samples, d and d', each of size m, which result from running algorithms a and a', respectively, m steps on some $f \in \mathcal{F}$. Based on these samples, the choosing procedure then decides which algorithm, a or a', to use on the subsequent part of the search of f.*

Also, for this particular NFL theorem, because we are comparing the choosing procedures, and not the algorithms, it is necessary to define a slightly different measure of performance. Remember that for the choosing procedure NFL theorem, we are saying that any two choosing procedures are equivalent. Thus, we are interested in what happens *after* a choosing procedure makes its choice. That is,

if the goal is to find the maximum, did the choosing procedure make a "good" decision and choose the algorithm that leads to larger cost values, or did the choosing procedure make a "poor" decision and choose the algorithm that leads to small cost values? The question of which choosing procedure made the better decision (and chose the better algorithm) is answered by comparing where each of the algorithms would explore after the initial m iterations. In other words, we are interested in $d_{>m}$, the samples of the function that come after using the choosing algorithm. Thus, a performance measure of interest is $\Phi(\vec{c}_{>m})$, the rank (goodness) of the histogram of cost values that comes after the initial m sample points (i.e., after the choosing procedure has made its choice).

2.2.2 Statement of Theorem

Now that we have all of the necessary framework in place, from [2], the NFL theorem involving choosing procedures is as follows:

Theorem 1 (Choosing Procedure NFL Theorem - Wolpert and Macready)**.** *Let d and d' be two fixed samples of size m, that are generated when the algorithms a and a', respectively, are run on the (arbitrary) cost function f at hand. Let A and B be two different choosing procedures. Let k be the number of elements in $\vec{c}_{>m}$. Then*

$$\sum_{a,a'} P(\vec{c}_{>m}|f,d,d',k,a,a',A) = \sum_{a,a'} P(\vec{c}_{>m}|f,d,d',k,a,a',B)$$

where the sum is over all algorithms a consistent with d and a' consistent with d' and $P(\vec{c}_{>m}|\ldots)$ is the probability that a certain histogram $\vec{c}_{>m}$ is produced for the given parameters.

An immediate result of this theorem is that

$$P_{a,a'}(\vec{c}_{>m}|f,d,d',k,a,a',A) = P_{a,a'}(\vec{c}_{>m}|f,d,d',k,a,a',B).$$

That is, for fixed f, d, d', and k, once a choosing procedure has made its choice, the probability, over uniform random choice of a and a' consistent with d and d', of seeing a particular histogram of y-values, $\vec{c}_{>m}$, is independent of the choosing procedure. To see that this is the case, assuming that the choosing procedure is A, notice that $P_{a,a'}(\vec{c}_{>m}|f,d,d',k,a,a',A)$ is just $\sum_{a,a'} P(\vec{c}_{>m}|f,d,d',k,a,a',A)$ divided by the number of (a,a') pairs consistent with d and d'. Since the number of (a,a') pairs consistent with d and d' does not depend on the choosing procedure, and we showed in Theorem 1 that $\sum_{a,a'} P(\vec{c}_{>m}|f,d,d',k,a,a',A)$ is independent of the choosing procedure, then $P_{a,a'}(\vec{c}_{>m}|f,d,d',k,a,a',A)$ does not depend on the choosing procedure either. Thus, when considering all algorithms a consistent with d and a' consistent with d', the probability, over choice of a and a' consistent with d and d', of seeing a certain set of y-values is unaffected by the particular choosing procedure. Since the probability of $\vec{c}_{>m}$ is unchanged by the particular choosing procedure, the performance of the procedure $\Phi(\vec{c}_{>m})$ is also unchanged. Thus, the performance of all choosing procedures is equivalent.

For a detailed proof of Theorem 1 see [2]. However, worthy of mention here is that the proof relies on a certain restriction of where the algorithms can search after the initial m iterations. In particular, once a choosing procedure is selected, the chosen algorithm is not allowed to return to any points in $d_{\cup} \equiv d_m \cup d'_m$. More accurately, it *can* return to these points, but as explained before, these points are not counted. This may seem a strange restriction because, in essence, it prevents one algorithm from visiting any of the points that the *other* algorithm has already visited. As we will see, no such restriction is necessary in the new NFL result.

Chapter 3

Extending the Choosing Procedure NFL Theorem

3.1 Description of Scenario

The new NFL result, presented in this paper, is an extension to the choosing procedure theorem described in the previous chapter. The NFL theorem from [2] involved choosing procedures that make their decisions based on performance on one run each of a pair of optimization algorithms. Also, once a decision has been made, performance is measured based on a continuation of one of the algorithm runs. For the new NFL theorem, the choosing procedures instead choose based on performance on multiple runs (a training set of runs) of a pair of algorithms, and once a choice has been made, performance is measured on a new algorithm run, starting from a new, before unseen, initial x-value.

Somewhat more formally, the choosing procedures examine the performance of two algorithms, a and a', on a set of N algorithm runs on some function $f \in \mathcal{F}$, where the initial x-values for each run are chosen uniformly at random. Then, based on this performance, a choosing procedure decides which algorithm, a or a', to use

on a new algorithm run, with initial x-value again chosen uniformly at random, however with the restriction that only a starting x-value not already seen before will be accepted. As the following theorem and proof will show, given this setting, any two choosing procedures are statistically equivalent.

3.2 New NFL Result

3.2.1 Additional Definitions

Before presenting our new result, we will first provide a slightly different definition for a choosing procedure since we now are making a decision based on a set of algorithm runs rather than on a single run. Also, we will explicitly include in our definition the possibility that a choosing procedure is randomized. This allows us to include a random choosing procedure in our subsequent analysis. The definition for a choosing procedure is as follows:

Definition 2 (Choosing Procedure)**.** *A* choosing procedure *is a rule that examines the samples, $d_1, d_2, ..., d_N$ and $d'_1, d'_2, ..., d'_N$, each of size m, which result from running algorithms a and a', respectively, N times on some $f \in \mathcal{F}$, where the initial x-value for each run is chosen uniformly at random; call these values $x_1, x_2, \ldots, x_N$. Based on these samples, the choosing procedure then decides which algorithm, a or a', to use on the $(N+1)$th algorithm run on f, where the starting x-value, x_{N+1}, for this run is again chosen uniformly at random, but with the restriction that $x_{N+1} \notin \{x_1, x_2, \ldots, x_N\}$. The choosing procedure can also elect to ignore the samples and randomly choose between a and a', with some fixed probability γ of choosing a and probability $1-\gamma$ of choosing a'.*

Also, we need to refine our definition of an algorithm since we are now

randomly choosing its initial x-value. Recall that the previous definition of an algorithm says that it chooses its own starting value: $a : d \in \mathcal{D} \mapsto \{x | x \notin d^x\}$, where $\mathcal{D} \equiv \cup_{m \geq 0} \mathcal{D}_m$ is the set of all samples of arbitrary size and $\mathcal{D}_m = (\mathcal{X} \times \mathcal{Y})^m$ is the space of all samples of size m. That an algorithm chooses its starting value is reflected in the fact that m can equal 0. For our new definition, we still want an algorithm to be defined as a mapping from a set of visited points to a new, previously unvisited, point. However, we need to reflect that we are choosing the algorithm's starting value. We can accomplish this by simply redefining $\mathcal{D}$ as $\mathcal{D} \equiv \cup_{m \geq 1} \mathcal{D}_m$, and leaving the definition of an algorithm as $a : d \in \mathcal{D} \mapsto \{x | x \notin d^x\}$.

3.2.2 Theorem and Proof

Now that an introduction to the scenario has been presented and we have redefined a choosing procedure and an algorithm to fit this scenario, a formal statement of the theorem and proof (which is an adaptation of Wolpert and Macready's proof of Theorem 1) are as follows:

Theorem 2 (New Choosing Procedure NFL Theorem)**.** *Let $d_1, d_2, ..., d_N$ and $d'_1, d'_2, ..., d'_N$ be fixed samples, each of size m, that are generated when two algorithms a and a', respectively, are run N times on an (arbitrary) fixed function $f \in \mathcal{F}$, where the initial x-value for each run is drawn uniformly at random; call these values $x_1, x_2, \ldots, x_N$. Let A and B be two different choosing procedures. Let x_{N+1} be the starting value for the $(N+1)$th algorithm run on f, where x_{N+1} is again chosen uniformly at random, but with the restriction that $x_{N+1} \notin \{x_1, x_2, \ldots, x_N\}$. Let m also be the number of elements in $\vec{c}_{>m \cdot N}$, where $\vec{c}_{>m \cdot N}$ is defined as the histogram of cost values that results after running the chosen algorithm until it has visited m distinct points. Then*

$$\sum_{a,a'} P(\vec{c}_{>m\cdot N}|f, x_{N+1}, d_1, d_2, \ldots, d_N, d'_1, d'_2, \ldots, d'_N, m, a, a', A)$$

$$= \sum_{a,a'} P(\vec{c}_{>m\cdot N}|f, x_{N+1}, d_1, d_2, \ldots, d_N, d'_1, d'_2, \ldots, d'_N, m, a, a', B)$$

where the sum is over all algorithms a consistent with $d_1, d_2, ..., d_N$ and a' consistent with $d'_1, d'_2, ..., d'_N$.

Proof. Let *proc* represent A or B. For deterministic choosing procedures, for fixed $d_1, d_2, \ldots, d_N$ and $d'_1, d'_2, \ldots, d'_N$, *proc*'s choice of a or a' is fixed. So, throughout the summation, a choosing procedure consistently chooses a or consistently chooses a' because it makes its choice based on $d_1, d_2, \ldots, d_N$ and $d'_1, d'_2, \ldots, d'_N$, and not on the particular algorithms at hand. This leaves us with two possible situations: 1. if *proc* chooses a; or 2. if *proc* chooses a'.

If *proc* chooses a for our $d_1, d_2, \ldots, d_N$ and $d'_1, d'_2, \ldots, d'_N$ then we have

$$\sum_{a,a'} P(\vec{c}_{>m\cdot N}|f, x_{N+1}, d_1, d_2, \ldots, d_N, d'_1, d'_2, \ldots, d'_N, m, a, a', proc)$$

$$= \sum_{a'} \Big(\sum_{a} P(\vec{c}_{>m\cdot N}|f, x_{N+1}, d_1, d_2, \ldots, d_N, d'_1, d'_2, \ldots, d'_N, m, a) \Big)$$

where a' can be dropped from $P(\vec{c}_{>m\cdot N}|\ldots)$ because $\vec{c}_{>m\cdot N}$ is set by a if a is chosen, and *proc* can be dropped because of its consistent choosing of a. Also, we can break our original (single) summation into two because fixing an a' and then an a gives the same result as picking an (a, a') pair since we are summing over all possible valid combinations of a and a' (i.e., all a and a' consistent with the data).

If *proc* chooses a' for our $d_1, d_2, \ldots, d_N$ and $d'_1, d'_2, \ldots, d'_N$ then we have

$$\sum_{a,a'} P(\vec{c}_{>m\cdot N}|f, x_{N+1}, d_1, d_2, \ldots, d_N, d'_1, d'_2, \ldots, d'_N, m, a, a', proc)$$

$$= \sum_{a} \Big(\sum_{a'} P(\vec{c}_{>m\cdot N}|f, x_{N+1}, d_1, d_2, \ldots, d_N, d'_1, d'_2, \ldots, d'_N, m, a') \Big)$$

where a and $proc$ can be dropped and the summation can be broken into two for reasons analogous to above.

Starting with the inner summations on each of the right-hand sides, by the definitions of $\vec{c}_{>m\cdot N}$ and x_{N+1}, on the $(N+1)$th runs of a and a', neither algorithm has seen a sample of length m with initial x-value x_{N+1}. That is, neither algorithm has seen the set of data that results when using the initial x-value x_{N+1} on their algorithm. To see that this is the case, recall that the definition of an algorithm is a mapping from a previously visited (ordered) set of points to a single new (previously unvisited) point in $\mathcal{X}$. So each step that an algorithm takes depends on the ordered set of previous steps it has taken. Because we have never started a and a' with an initial x-value of x_{N+1}, the algorithms are unconstrained. That is, the algorithms are not forced into taking the same steps they previously took on one of the runs in the training set. So, despite the fact that the algorithms have already been run N times on f, what the algorithms do on the $(N+1)$th run (i.e., the test run) is independent of what they did the first N times (i.e., independent of the samples $d_1, d_2, \ldots, d_N$ and $d'_1, d'_2, \ldots, d'_N$ that came from running the algorithms a and a', respectively, the first N times). Thus, both algorithms are free to visit any possible sequence of m x-values beginning with x_{N+1}. Therefore, the inner summations in the above equations are equal because they both sum over the same set of possibilities for $\vec{c}_{>m\cdot N}$.

Because $P(\vec{c}_{>m\cdot N}|\ldots, a)$ has no dependence on a' and $P(\vec{c}_{>m\cdot N}|\ldots, a')$ has no dependence on a, in order to show that the overall sums of the right-hand sides

of the equations are equal, we only need to show that each outer summation is added up the same number of times. That is, the number of possible a's and the number of possible a''s is the same. To see that this is true, notice that in our summation we are implicitly summing over those a and a' that could result in $d_1, d_2, \ldots, d_N$ and $d'_1, d'_2, \ldots, d'_N$ respectively when run on f. Before the first runs of the optimization algorithms on f, both a and a' could come from the set of all possible algorithms. Then, as each set of algorithm runs is performed, each set of data points $d_1, d_2, \ldots, d_N$ and $d'_1, d'_2, \ldots, d'_N$ puts constraints on which algorithms a and a' could be. However, each set of points puts the same number and kind of constraints on both a and a'. That is, each knocks out N (or some number less than N if any of the x_i for $i = 1, 2, \ldots, N$ are the same) sequences of m initial values. So it does not matter what the data is, the same number of algorithms are taken out of consideration as a possible algorithm at each step. Thus, after the N runs of the optimization algorithm, the number of a's consistent with $d_1, d_2, \ldots, d_N$ and the number of a''s consistent with $d'_1, d'_2, \ldots, d'_N$ is the same. Therefore, the number of times that each outer summation is added is the same.

Thus, the overall sums are equal because for the inner summations a is summed over a set of values and a' is summed over (essentially) the same set, and the number of summands is equal in both cases. Another way to view this is to notice that we essentially have a change of variables, where although the order of summation may be different, each sum involves the same set of possible x values, which in turn will result in the same set of probability values, $P(\vec{c}_{>m\cdot N}|\ldots)$, being added. And again, because the number of outer summations is the same, this set of probability values is being added the same number of times. Thus, regardless of what *proc* does, the sums are the same.

Letting C equal the number of times the outer summation is summed, and assuming, without loss of generality, that *proc* chooses a, we can rewrite our sum as

$$\sum_{a,a'} P(\vec{c}_{>m\cdot N}|f, x_{N+1}, d_1, d_2, \ldots, d_N, d'_1, d'_2, \ldots, d'_N, m, a, a', proc)$$

$$= C \cdot \sum_{a} P(\vec{c}_{>m\cdot N}|f, x_{N+1}, d_1, d_2, \ldots, d_N, d'_1, d'_2, \ldots, d'_N, m, a).$$

This sum is independent of *proc*, which in turn allows us to say that the sum for choosing procedure A is equal to the sum for choosing procedure B, or more precisely

$$\sum_{a,a'} P(\vec{c}_{>m\cdot N}|f, x_{N+1}, d_1, d_2, \ldots, d_N, d'_1, d'_2, \ldots, d'_N, m, a, a', A)$$

$$= \sum_{a,a'} P(\vec{c}_{>m\cdot N}|f, x_{N+1}, d_1, d_2, \ldots, d_N, d'_1, d'_2, \ldots, d'_N, m, a, a', B).\square$$

Note that in the proof of this theorem choosing procedures A and B are assumed to be deterministic. This is reflected in our claim that for fixed samples d_1,$d_2, \ldots, d_N$ and $d'_1, d'_2, \ldots, d'_N$ a choosing procedure's choice of a or a' is fixed. Since our definition of a choosing procedure also allows for a form of randomization in which the samples are ignored and an algorithm is randomly selected, we need to extend Theorem 2 with the following corollary.

Corollary 1. *For randomized choosing procedures A and B, and for $\vec{c}_{>m\cdot N}, f, x_{N+1}, d_1, d_2, \ldots, d_N, d'_1, d'_2, \ldots, d'_N, m, a,$ and a' as defined in Theorem 2,*

$$\sum_{a,a'} P(\vec{c}_{>m\cdot N}|f, x_{N+1}, d_1, d_2, \ldots, d_N, d'_1, d'_2, \ldots, d'_N, m, a, a', A)$$

$$= \sum_{a,a'} P(\vec{c}_{>m\cdot N}|f, x_{N+1}, d_1, d_2, \ldots, d_N, d'_1, d'_2, \ldots, d'_N, m, a, a', B).$$

Proof. For a fixed setting of the parameters (f, d's, d''s, etc.), a randomized choosing procedure R is going to choose a with some probability γ and a' with some probability $(1-\gamma)$. So for a fixed a and a', $P(\vec{c}_{>m\cdot N}|\ldots,R)$ is γ for the $\vec{c}_{>m\cdot N}$ produced by a, $(1-\gamma)$ for the $\vec{c}_{>m\cdot N}$ produced by a', and 0 for all other histograms $\vec{c}_{>m\cdot N}$. Let A be the procedure that always chooses a and let A' be the procedure that always chooses a'. Then summing $P(\vec{c}_{>m\cdot N}|\ldots,R)$ over all (a,a') pairs, you get γ times the sum of the probabilities $P(\vec{c}_{>m\cdot N}|\ldots,A)$ and $(1-\gamma)$ times the sum of the probabilities $P(\vec{c}_{>m\cdot N}|\ldots,A')$. However, since A and A' are deterministic, the proof of Theorem 2 gives you that the sum of the probabilities $P(\vec{c}_{>m\cdot N}|\ldots,A)$ equals the sum of the probabilities $P(\vec{c}_{>m\cdot N}|\ldots,A')$. Therefore,

$$\begin{aligned}
\sum_{a,a'} P(\vec{c}_{>m\cdot N}|\ldots,R) &= \gamma\cdot\sum_{a,a'} P(\vec{c}_{>m\cdot N}|\ldots,A)+(1-\gamma)\cdot\sum_{a,a'} P(\vec{c}_{>m\cdot N}|\ldots,A') \\
&= \gamma\cdot\sum_{a,a'} P(\vec{c}_{>m\cdot N}|\ldots,A)+(1-\gamma)\cdot\sum_{a,a'} P(\vec{c}_{>m\cdot N}|\ldots,A) \\
&= \sum_{a,a'} P(\vec{c}_{>m\cdot N}|\ldots,A).\square
\end{aligned}$$

Thus, Theorem 2 can be extended to show that there is still no free lunch for randomized choosing procedures. An immediate result of this theorem (and corollary) is that

$$P_{a,a'}(\vec{c}_{>m\cdot N}|f,x_{N+1},d_1,d_2,\ldots,d_N,d'_1,d'_2,\ldots,d'_N,m,a,a',A)$$

$$=P_{a,a'}(\vec{c}_{>m\cdot N}|f,x_{N+1},d_1,d_2,\ldots,d_N,d'_1,d'_2,\ldots,d'_N,m,a,a',B).$$

That is, for fixed f, x_{N+1}, d's, d''s, and m, the probability over uniform random choice of a and a' of seeing a particular histogram of y-values, $\vec{c}_{>m\cdot N}$, is independent of the choosing procedure. In other words, when considering all algorithms a consistent with the d's and a' consistent with the d''s, the probability, over uniform

choice of a and a' consistent with the d's and d''s, of seeing any particular set of y-values does not depend on the choosing procedure. Thus, when averaging across all pairs of algorithms consistent with the data, all choosing procedures are equivalent.

Theorem 2 leads to the following corollary:

Corollary 2. *Given Theorem 2, the following holds:*

$$E_{a,a',x_{N+1}}\left[\Phi(\vec{c}_{>m\cdot N})|f, x_{N+1}, d_1, d_2, \ldots, d_N, d'_1, d'_2, \ldots, d'_N, m, a, a', A\right]$$

$$= E_{a,a',x_{N+1}}\left[\Phi(\vec{c}_{>m\cdot N})|f, x_{N+1}, d_1, d_2, \ldots, d_N, d'_1, d'_2, \ldots, d'_N, m, a, a', B\right]$$

where the expectation is over the uniform random choice of starting point x_{N+1} and uniform random choice of a and a' consistent with the training data, and A and B are any two choosing procedures.

Proof. To see that this is true, assume A is the choosing procedure. For fixed $f, x_{N+1}, d_1, d_2, \ldots, d_N, d'_1, d'_2, \ldots, d'_N$, and m, the expected value, over all possible (a, a') pairs consistent with the data, of the goodness of a histogram, $\Phi(\vec{c}_{>m\cdot N})$, given we are using choosing procedure A, is just the sum, over all possible histograms $\vec{c}_{>m\cdot N}$, of the goodness of that histogram times the probability over a and a' of obtaining the histogram given A. That is,

$$E_{a,a'}\left[\Phi(\vec{c}_{>m\cdot N})|f, x_{N+1}, d_1, d_2, \ldots, d_N, d'_1, d'_2, \ldots, d'_N, m, a, a', A\right]$$

$$= \sum_{\vec{c}_{>m\cdot N}} \Phi(\vec{c}_{>m\cdot N}) \cdot P_{a,a'}(\vec{c}_{>m\cdot N}|f, x_{N+1}, d_1, d_2, \ldots, d_N, d'_1, d'_2, \ldots, d'_N, m, a, a', A).$$

This probability, $P_{a,a'}(\vec{c}_{>m\cdot N}|\ldots, a, a', A)$, is just the sum over a and a' of $P(\vec{c}_{>m\cdot N}|\ldots, a, a', A)$ divided by the number of (a, a') pairs consistent with

$d_1, d_2, \ldots, d_N$ and $d'_1, d'_2, \ldots, d'_N$. We will use $\#(a, a')$ to denote this number. Thus we have

$$E_{a,a'}\left[\Phi(\vec{c}_{>m\cdot N})|f, x_{N+1}, d_1, d_2, \ldots, d_N, d'_1, d'_2, \ldots, d'_N, m, a, a', A\right]$$

$$= \sum_{\vec{c}_{>m\cdot N}} \Phi(\vec{c}_{>m\cdot N}) \cdot \frac{\sum_{a,a'} P(\vec{c}_{>m\cdot N}|f, x_{N+1}, d_1, d_2, \ldots, d_N, d'_1, d'_2, \ldots, d'_N, m, a, a', A)}{\#(a, a')}.$$

In Theorem 2 we showed that $\sum_{a,a'} P(\vec{c}_{>m\cdot N}|\ldots, a, a', A)$ is independent of the choosing procedure. Since no other terms involve A, the choosing procedure, we get that $E_{a,a'}\left[\Phi(\vec{c}_{>m\cdot N})|\ldots, a, a', A\right]$ is also independent of the choosing procedure. Thus we have

$$E_{a,a'}\left[\Phi(\vec{c}_{>m\cdot N})|f, x_{N+1}, d_1, d_2, \ldots, d_N, d'_1, d'_2, \ldots, d'_N, m, a, a', A\right]$$

$$= E_{a,a'}\left[\Phi(\vec{c}_{>m\cdot N})|f, x_{N+1}, d_1, d_2, \ldots, d_N, d'_1, d'_2, \ldots, d'_N, m, a, a', B\right].$$

To complete the proof, we need to take the expected value over x_{N+1} as well. By definition of expected value, we have

$$E_{a,a',x_{N+1}}\left[\Phi(\vec{c}_{>m\cdot N})|f, x_{N+1}, d_1, d_2, \ldots, d_N, d'_1, d'_2, \ldots, d'_N, m, a, a', A\right]$$

$$= \sum_{x_{N+1}} E_{a,a'}\left[\Phi(\vec{c}_{>m\cdot N})|f, x_{N+1}, d_1, d_2, \ldots, d_N, d'_1, d'_2, \ldots, d'_N, m, a, a', A\right] \cdot P(x_{N+1}).$$

The only term which involves the choosing procedure A is $E_{a,a'}\left[\Phi(\vec{c}_{>m\cdot N})|\ldots, x_{N+1}, a, a', A\right]$ which we have already shown is independent of the choosing procedure. Thus

$$E_{a,a',x_{N+1}}\left[\Phi(\vec{c}_{>m\cdot N})|f, x_{N+1}, d_1, d_2, \ldots, d_N, d'_1, d'_2, \ldots, d'_N, m, a, a', A\right]$$

$$= E_{a,a',x_{N+1}}\left[\Phi(\vec{c}_{>m\cdot N})|f, x_{N+1}, d_1, d_2, \ldots, d_N, d'_1, d'_2, \ldots, d'_N, m, a, a', B\right].\square$$

This corollary says that for any fixed training data, the expected performance of all choosing procedures is identical when averaged over all possible unseen starting points and over all optimization algorithms consistent with the data. This would seem to imply that no choosing procedure can have expected performance—over choice of starting point and algorithms—better than a random choosing procedure.

Also, from this result, Wolpert and Macready would claim that regardless of f, if we make no assumptions about how a and a' are chosen, then we cannot be sure that any particular choosing procedure will perform better than a random choosing procedure when the test point is chosen at random from the set of unseen starting points. This in turn implies that, barring assumptions about the optimization algorithms and/or f, there is no theoretical justification for using any particular choosing procedure. In the next chapter we will show that this is not necessarily the case.

Chapter 4

A Superior Choosing Procedure

One of the conclusions that might be drawn from Theorem 2 is that there are no "superior" choosing procedures. This is because this theorem tells us that, mathematically, "good" performance is offset by "bad" performance, so on average no choosing procedure is any better than any other choosing procedure. Put another way, this result implies that no choosing procedure is superior to a procedure that randomly selects an algorithm because instances in which a choosing procedure does better than random are seemingly offset by instances in which it does worse than random.

Using probability theory in conjunction with principles from standard cryptographic practice we will show that instances in which a choosing procedure does better than random do not *necessarily* have to be offset by instances in which it does worse than random. More specifically, we will present an example of a choosing procedure that, if provided with a sufficient (yet reasonable) number of training runs, we should rationally believe performs at least as well as and sometimes better than a random choosing procedure. That is, we should rationally believe that, on average, the performance of this choosing procedure is superior to a random chooser and thus it can be classified as a general-purpose choosing procedure.

4.1 Description of Choosing Procedure

This "superior" choosing procedure makes its choice as follows: if one of a pair of algorithms outperforms the other on *all* algorithm runs in the training set, then choose this algorithm; otherwise, randomly choose between the two algorithms, where each algorithm is chosen with probability 1/2. We will refer to this choosing procedure as the *unanimous choosing procedure* since it only really makes a decision if one algorithm *always* outperforms the other one (i.e., when there is unanimous support for one of the algorithms). In Section 4.2.5 a formal definition will be provided.

4.2 Why the Procedure Is Superior

In order to show that the unanimous choosing procedure is superior we must determine the "sufficient, yet reasonable" number of training runs that are necessary to claim that the procedure will perform better than random choosing, on average. First we will provide some intuition as to why a sufficiently large training set allows us to make such a claim. The intuition is as follows. If one algorithm consistently outperforms the other for all N runs in the training set, then, using standard probability theory, we can show that the probability that the unanimous choosing procedure will be fooled into thinking that this algorithm is better (when in reality the other algorithm is better) becomes increasingly small as N grows. If we can say that the probability that our choosing procedure is fooled in such a way is so small—perhaps around 2^{-128} small—that we can rationally believe that it will not happen, this allows us to say that our choosing procedure is better than random guessing. This is because, on one set of possible test runs, when one algorithm does not consistently outperform the other during the training runs, our choosing procedure reduces to random guessing. On the other set of possible test runs, when

one algorithm consistently does better, then it is overwhelmingly likely, and thus rational for us to believe, that on average the algorithm that it chooses performs better on the test run. So, on one set of runs our choosing procedure does as well as random, and on the other set we can rationally believe that it does better than random. This allows us to say that, on average, our choosing procedure is better that random guessing, or alternatively, that it is a general-purpose choosing procedure.

4.2.1 Prediction Error of a Choosing Procedure

Using a somewhat more precise argument, notice that when a random choosing procedure selects an algorithm, then with probability 1/2 this choice is correct, that is, the chosen algorithm will perform better on the test run, and with with probability 1/2 this choice is incorrect. Thus, we will say that a random choosing procedure's *prediction error* is 1/2.

For the unanimous choosing procedure, when one algorithm does *not* consistently outperform the other, then it randomly selects an algorithm. From the preceding discussion, we see that when this happens its prediction error is 1/2. When one algorithm *does* consistently outperform the other and the number of times that it does so is sufficiently large, then we can show that it is rational to believe that the prediction error is less than 1/2. Thus, averaged over unseen starting values, it is rational to believe that the prediction error of the unanimous choosing procedure is less than 1/2.

The NFL result of Chapter 3 seems to imply that the expected prediction error is exactly 1/2 for all choosing procedures. Thus, if we can show that it is rational to believe that the expected prediction error of the unanimous choosing procedure is less than 1/2, then we can say that the implications of the NFL theorem are not as negative as expected.

4.2.2 A Sufficient Training Set

Now that we have some intuition on why a sufficiently large training set allows us to claim that our choosing procedure is superior to random guessing, we can start to pin down an an appropriate value for N. We can do this using standard probability theory.

As previously discussed, when the unanimous choosing procedure selects an algorithm (i.e., does not randomly choose) we need the prediction error to be less than 1/2 in order to claim this procedure is better than random. That is, when the procedure makes a choice, the probability over unseen starting values that the chosen algorithm loses, or performs worse than the other algorithm, must be less than 1/2. Using standard probability theory we can present a situation in which it is overwhelmingly likely that a certain *classification error* ϵ_c holds, where classification error is the probability over *all possible* starting values that the chosen algorithm performs worse using the starting value. As we will show in Section 4.2.3, we can then use this classification error to calculate the prediction error.

Since we can show it is extremely likely that a particular classification error ϵ_c holds, let us fix this value to $\epsilon_c = 0.24$. This way, even if the prediction error is double the classification error we still have $\epsilon_p = 0.48 < \frac{1}{2}$. As we will show in Section 4.2.3, if the number of training runs is less than $\frac{1}{2}|\mathcal{X}|$ this is a fitting assumption. We can now begin to calculate a training set size N such that it is overwhelmingly likely, and thus rational to believe, that the classification error is at most 0.24 (and after some calculations, that the prediction error is at most 0.48).

This argument is as follows. If the probability is at least 0.24 that the chosen algorithm loses (i.e., the classification error is at least 0.24), then on one training run the probability that the unanimous choosing procedure does not pick the "losing" algorithm, or alternatively, the probability that it fails to detect a loss is at most $1 - 0.24 = 0.76$. On N training runs, the probability that it fails to detect any losses

is at most $(1-0.24)^N = (0.76)^N$. This is because training runs are independent, so we can simply multiply the probabilities. Thus, on the $(N+1)$th run or the test run of the algorithms, the probability that the unanimous choosing procedure is "fooled" by the randomized choice of starting values in the training set into thinking that one algorithm is better 76% of the time (because it did not detect any losses during the training runs) and thus is "fooled" into choosing this algorithm is at most $(0.76)^N$. If we set this value, $(0.76)^N$, less than some extraordinarily small value $\delta > 0$ and solve for N, then it is extremely unlikely that the unanimous choosing procedure will be fooled by the N training runs into choosing the algorithm that will perform worse on the test run. In fact, in Section 4.2.6 we claim that if the probability that it is fooled is less than δ, then it is rational for us to believe that it will never be fooled. Thus, if we encounter a situation in which one algorithm consistently wins (for a sufficient number of training runs), then it must be the case that we are not being fooled but instead happen to have a "nice" function that is not completely random or happen to have algorithms that can truly distinguish themselves which means that we should trust the selection that the choosing procedures makes.

In order to solve for the sufficient training set size N such that it is extremely unlikely that the unanimous choosing procedure is fooled into making an incorrect choice, we will set the extraordinarily small value δ to $2^{-\sigma}$, where for the time being and for concreteness we will use $\sigma = 128$. That is, we will use $\delta = 2^{-\sigma} = 2^{-128}$. As we will see in Section 4.2.7, the reason for choosing this as our δ and in particular for choosing 128 as our σ is related to standard cryptographic practice. For now we will just note that 2^{-128} is an extraordinarily small probability.

In order to find a sufficient training set size N such that $(0.76)^N < \delta$ we can use the following formula from [4]: $N \geq \left\lceil \frac{1}{\epsilon_c} ln(\frac{1}{\delta}) \right\rceil$. Note that $(0.76)^N$ is actually just $(1-\epsilon_c)^N$, so $\epsilon_c = 0.24$. To verify that this formula is correct and to see why values of N that satisfy this inequality are appropriate, we use the fact that

$(1-\frac{1}{n})^n < \frac{1}{e}$ for $n > 0$. If we substitute ϵ_c in for $\frac{1}{n}$ and raise both sides to the power $ln(\frac{1}{\delta})$ this becomes $(1-\epsilon_c)^{(\frac{1}{\epsilon_c})(ln\frac{1}{\delta})} < (\frac{1}{e})^{ln\frac{1}{\delta}} = \frac{1}{\frac{1}{\delta}} = \delta$. Thus, if we set N at least as large as $(\frac{1}{\epsilon_c})(ln\frac{1}{\delta})$, we get $(1-\epsilon_c)^N < \delta$. So for $\epsilon_c = 0.24$ and $\delta = 2^{-128}$, we have $\left\lceil \frac{1}{\epsilon_c} ln(\frac{1}{\delta}) \right\rceil = \left\lceil \frac{1}{0.24} ln(\frac{1}{2^{-128}}) \right\rceil = 370$. Thus, when the unanimous choosing procedure makes a choice, that is, does not randomly pick an algorithm, values of N greater than or equal to 370 will allow us to produce an algorithm choice that with probability $(1-2^{-128})$ will lose at most 24% of the time (i.e., have classification error at most 0.24).

Notice that this says that the *classification error* is at most 0.24. That is, 0.24 is the probability, taken over *all* $x \in \mathcal{X}$, that the unanimous choosing procedure selects the losing algorithm when using starting value x. This means that x's seen during the training runs are used in the calculation of this probability. We are interested in how the choosing procedure performs on $x_{N+1} \notin \{x_1, x_2, \ldots, x_N\}$, or how it performs on *unseen* x, so we are interested in the *prediction error*. In the next section we will show how to use classification error to calculate prediction error.

4.2.3 From Classification to Prediction Error

We can calculate the prediction error from the classification error as follows. If the classification error rate of the unanimous choosing procedure, when it selects an algorithm, is 0.24, then it makes at most $0.24 \cdot |\mathcal{X}|$ errors on the entire space $\mathcal{X}$. If we have seen some fraction α of the possible starting x-values, then even if all of the errors are on unseen starting values, the prediction error rate ϵ_p is at most (number of errors)/(number of unseen inputs) $= \frac{0.24 \cdot |\mathcal{X}|}{(1-\alpha) \cdot |\mathcal{X}|} = \frac{0.24}{1-\alpha}$. So we now have a bound of $\frac{0.24}{1-\alpha}$ on the prediction error rate.

In order to show that the overall prediction error rate of the unanimous choosing procedure is less than 1/2, and thus better than what Theorem 2 would seem to imply, we need the prediction error when a choice is made to be less than

1/2. Thus, we need to solve for α to determine the allowable fraction of seen starting values such that this prediction error is less than 1/2. That is, we need to solve $\frac{0.24}{1-\alpha} < \frac{1}{2}$ for α. Doing this, we see that α must be less than 0.52, so in order to ensure a prediction error of less than 1/2 we must limit the number of seen starting values to approximately $\frac{1}{2}|\mathcal{X}|$. Note that in practice it highly unlikely that we will sample half of the search space because $|\mathcal{X}|$ is typically very large.

From these calculations it should now be clear why a classification error of 0.24 was chosen. This is because if we limit the number of seen inputs to 1/2 of the search space, then the prediction error is $\frac{0.24}{1-\alpha} = \frac{0.24}{1-\frac{1}{2}} = 0.48$ which is less than 1/2. A classification error of even 0.25 would yield a prediction error of 0.5 which is not *less* than 1/2 and thus would not allow us to claim that the unanimous choosing procedure is better. Also, note that the prediction error we have calculated holds for the same N and with the same probability that the classification error holds, that is, it holds with probability $1 - 2^{-128}$ for $N = 370$.

4.2.4 A General-Purpose Choosing Procedure

Throughout this paper we have alternately discussed our choosing procedure as being better than random and as being a general-purpose choosing procedure. We will now provide a formal definition for a general-purpose choosing procedure which will more precisely describe what we mean by "general-purpose." Also, from this definition it should be clear that when we say a choosing procedure is a general-purpose choosing procedure this means that it is better than a random chooser.

Definition 3 (General-Purpose Choosing Procedure)**.** *A choosing procedure $\mathcal{P}$ is a* general-purpose choosing procedure *if every time it is executed on some fixed function f, the probability over unseen starting values $U \equiv \mathcal{X} - \{x_1, x_2, \ldots, x_N\}$ that*

the algorithm selected by $\mathcal{P}$ performs worse than the algorithm that was not selected is less than or equal to $1/2$ *and sometimes strictly less than* $1/2$*. That is, its prediction error* ϵ_p *is less than or equal to* $1/2$*, and in some cases is strictly less than* $1/2$*.*

Note that in this definition we are saying that there exist situations where ϵ_p is *strictly less than* $1/2$. In Section 4.2.9 we will provide one such situation. Also note that from this definition it should be clear that a general-purpose choosing procedure is better than random. This is because a random chooser always has $\epsilon_p = 1/2$; however, as we have already stated, there are instances in which a general-purpose choosing procedure has $\epsilon_p < 1/2$. Thus, the average prediction error of a general-purpose choosing procedure will be less than $1/2$, and thus better than random guessing. Finally, we use the term "general-purpose" because *every time* the procedure is executed the probability that the selected algorithm performs worse is at most $1/2$ (and sometimes less). So no matter when we use it, it will perform at least as well as and sometimes better than random. Thus, we can consider it to be "general-purpose."

4.2.5 The Unanimous Choosing Procedure

Now that we have the necessary framework in place, we can finally give a formal definition of the unanimous choosing procedure. This will then allow us to formalize our argument that this procedure is better than random and is a general-purpose choosing procedure.

Definition 4 (Unanimous Choosing Procedure)**.** *The* unanimous choosing procedure *is a choosing procedure that runs the algorithms a and a' on a sufficient number of training set starting x-values, N, such that the probability over choice of*

starting values $S = \{x_1, x_2, \ldots, x_N\}$ *that either* a *is correct on every* x_i *or* a' *is correct on every* x_i *and yet the procedure has classification error greater than* 0.24 *is at most* $2^{-\sigma}$*. Once the sufficient number of training runs is performed, the procedure then makes its choice as follows. If* a *is perfectly correct on* S *or* a' *is perfectly correct on* S *and* $N \leq \frac{1}{2}|\mathcal{X}|$*, it selects* a *or* a'*, respectively; otherwise it randomly chooses between* a *and* a'*, where each algorithm has probability* 1/2 *of being chosen.*

Note that in the above definition the choosing procedure will only make a choice if one algorithm always wins *and* if it has seen at most 1/2 of all possible initial x-values. This is indicated by the restriction of N to $N \leq \frac{1}{2}|\mathcal{X}|$. As previously mentioned, given that the classification error ϵ_c is 0.24 we must limit the fraction of seen inputs to 1/2 in order to ensure that the prediction error ϵ_p is at most $2 \cdot 0.24 = 0.48$, which is less than 1/2.

4.2.6 Choosing Procedures and Rationality

Now that we have a more formal definition of the unanimous choosing procedure, we can discuss why it is rational to always trust that its prediction error is less than 1/2. Using terminology similar to [5], we define a class $\mathcal{TC}$ of *trustworthy* choosing procedures as follows:

Definition 5 (Trustworthy Class of Choosing Procedures)**.** *A choosing procedure* $\mathcal{P}$ *is in the class* $\mathcal{TC}$ *if for every execution of* $\mathcal{P}$ *(regardless of* $f, x_{N+1}, m, a, a', N, d$*'s, and* d'*'s), the probability, over uniform random choice of initial* x*-values from* U*, that the prediction error rate exceeds* 1/2 *is at most* $2^{-\sigma}$*.*

As previously noted, based on standard cryptographic practice, we will use $\sigma = 128$ in defining our sufficiently small probability. We now make the following

claim:

Claim 1. *If choosing procedure $\mathcal{P}$ is in the class $\mathcal{TC}$, then $\mathcal{P}$ is a general-purpose choosing procedure. That is, every time $\mathcal{P}$ is executed its prediction error will be at most* $1/2$ *and for some problem parameters will be less than* $1/2$.

Thus we are claiming that $\delta = 2^{-\sigma} = 2^{-128}$ is such an extraordinarily small number that the prediction error of the choosing procedure *will* be at most $1/2$. That is, we are no longer saying that with probability $(1 - \delta)$ the prediction error rate will be at most $1/2$, we are instead saying that we are *convinced* that the prediction error rate will be at most $1/2$. Thus, $\delta = 2^{-128}$ is such a small probability that we can safely ignore it, or put another way, the probability that the procedure is fooled by the randomized choice of starting values into choosing the wrong algorithm is so small it should be ignored.

This leads to the following rational belief. If we believe Claim 1 (i.e., if we believe that a probability of 2^{-128} is sufficiently small that it is negligible, which cryptographic practice would lead us to believe), then we should trust that the prediction error of the unanimous choosing procedure is at most $1/2$ and sometimes less.

4.2.7 Cryptographic Practice and Rationality

As mentioned several times previously, the basis of our using 2^{-128} as an appropriately small probability comes from standard cryptographic practice. As shown in [6] and [7], the current recommendation of the National Security Agency (NSA) of the United States Government is to use the Advanced Encryption System cryptographic algorithm with 128-bit keys (AES-128) to encrypt classified documents. For our purposes, the details of how AES-128 works are unimportant.

What is important however is that the algorithm relies on the belief that the probability is 2^{-128} that a single random guess of a key will decrypt an encrypted document. As Jackson and Tamon mention in [5], even if the same key was used to encrypt every classified document and a billion documents were encrypted per second for a billion years, if we systematically guessed and checked distinct keys for each document, the probability of any of these guesses succeeding would be less than 1 in 10 trillion.

It is unlikely that the national government would go about encrypting classified documents with AES-128 if they knew it was possible to "break" the algorithm in any reasonable amount of time. Thus, it is implicit that the NSA and hence cryptographic experts believe that it is rational to trust that any document encrypted using AES-128 will not be broken by a limited number of guesses, even though there is an extremely small probability that this could occur for any given document. Or put another way, they feel that it is rational to believe that real-world events that have an extraordinarily small probability of occurring will not occur, even though mathematically we cannot rule out their possibility [5].

Before moving on, we note that in our definitions we used $2^{-\sigma}$ rather that the fixed value of 2^{-128}. This allows us to adjust our sufficiently small probability to changes in standard cryptographic practice. Thus, if 128 bits of security is no longer considered sufficient to protect classified documents, then we can simply adjust σ to be some larger value that is acceptable.

Now that we have shown how the extraordinarily small probability of $2^{-\sigma}$ was chosen, we assert that if it is rational to trust that the current cryptographic standards used to encrypt classified documents are secure, then it is rational to believe Claim 1. That is, if we believe that AES-128 will not be broken, then we should believe that every time the unanimous choosing procedure is run the prediction error will be at most $1/2$ and in some cases less than $1/2$.

4.2.8 A Trustworthy Choosing Procedure

Now that we have given some arguments for why Claim 1 should be accepted, we will subsequently use it to show that there are indeed positive results regarding the ability of the unanimous choosing procedure without the need for any assumptions about what the function f, starting value x_{N+1}, or algorithms a and a' consistent with the data are. In particular, we will show that the unanimous choosing procedure is in the class $\mathcal{TC}$. Thus, we should trust that its prediction error rate truly is less than or equal to 1/2 and sometimes strictly less than 1/2.

Theorem 3. *The unanimous choosing procedure is in $\mathcal{TC}$.*

Proof. By its definition the unanimous choosing procedure either randomly chooses between a and a', each with probability 1/2, or it makes a choice of a or a'. When it randomly chooses the prediction error rate is always 1/2. This is because, by definition, the random choice has probability 1/2 of picking a and probability 1/2 of picking a'. Thus, when a random choice is made (with probability 1) we have an error rate of 1/2. When the choosing procedure makes a choice, that is selects a or selects a', then (by its definition) with probability at least $1 - 2^{-\sigma}$ the prediction error is less than of equal to and some times strictly less than 1/2 . Thus, overall, with probability at least $1 - 2^{-\sigma}$, the unanimous choosing procedure's prediction error is at most 1/2 and sometimes less meaning it is in $\mathcal{TC}$.□

This leads immediately to the following corollary:

Corollary 3. *If Claim 1 is accepted, then the unanimous choosing procedure is a general-purpose choosing procedure.*

Before giving these results we stated that we make no assumptions about f, x_{N+1}, or the algorithms a and a' that the unanimous choosing procedure uses. However, Corollary 2 in Section 3.2.2 tells us that if we make no assumptions about what x_{N+1}, a, and a' are, then

$$E_{a,a',x_{N+1}}\left[\Phi(\vec{c}_{>m\cdot N})|\ldots,a,a',A\right] = E_{a,a',x_{N+1}}\left[\Phi(\vec{c}_{>m\cdot N})|\ldots,a,a',B\right],$$

where the expectations are over the uniform random choice of starting value x_{N+1} and algorithms a and a' consistent with the training data, and A and B are any two choosing procedures. Thus, the expected performance over uniform random choice of x_{N+1}, a, and a' is independent of the choosing procedure. So the NFL result seems to imply (and the pioneer of the NFL theorems David H. Wolpert would say) that if we do not make some assumption about f, x_{N+1}, a and a', then no choosing procedure is any better than a random chooser. That is, we must be making some assumption about f or about how x_{N+1}, a, and a' are chosen—they cannot be chosen uniformly—if we have a choosing procedure that does better than random choosing.

However we are saying that our choosing procedure does better than random choosing *and* we make no assumptions about the function f, the starting x-value x_{N+1}, or about the algorithms a and a' that the unanimous choosing procedure uses. That is, regardless of f and regardless of the x_{N+1}, a, and a' that we pick, the unanimous choosing procedure does better than random choosing. Thus, despite the seemingly negative implications of the NFL theorems, if we believe Claim 1, then the actual ramifications are not quite so negative.

Note that we are not saying that the unanimous choosing procedure is necessarily a "good" choosing procedure, we are just saying that it is *better* than a random chooser. That is, its average prediction error is better than 1/2. This is

because if we ever have the case that one algorithm always outperforms the other, then it is rational to believe that the unanimous choosing procedure will pick the algorithm that performs better, on average, on test runs (i.e., it will choose the correct algorithm). So on one set of possible runs, when the unanimous choosing procedure actually picks an algorithm, the prediction error is at most 1/2 (and sometimes strictly less than 1/2), and on the other set of possible runs, when it randomly chooses an algorithm, its prediction error is exactly 1/2. So, on average, the prediction error of the unanimous choosing procedure is better than 1/2.

Thus, despite the NFL result proven in Section 3.2.2 which tells us that, on average, the prediction error of a choosing procedure is 1/2, if we believe Claim 1, then we can show that there does exist (at least) one choosing procedure (i.e., the unanimous choosing procedure) such that, on average, its prediction error is better than 1/2. Therefore, the implications of the NFL theorems are not as negative as expected.

Note that this statement relies on our belief in Claim 1, which we already argued in support of in Sections 4.2.6 and 4.2.7. It also relies on the fact that it is possible that one algorithm could always outperform the other algorithm. Thus at some point we could have a prediction error of less than 1/2 which means that the *average* prediction error is also less than 1/2. We will now show one possible scenario in which one algorithm would always outperform the other algorithm.

4.2.9 Scenario in Which One Algorithm Is Consistently Better

One possible situation where one algorithm would always outperform the other is as follows. Recall that $\Phi(\vec{c}_{>m\cdot N})$ is our performance measure. One possible Φ that would give non-trivial results (i.e., the unanimous choosing procedure would make a choice) would be one that looks at the final y-value of a training run and

outputs 1 if this value is above a certain threshold and 0 otherwise. If used in conjunction with a convex function f (or a "bowl terrain") which takes on integer values from, say, 0 to 100, then our threshold might be something like 50. If algorithm a is some kind of hill-climber and algorithm a' is some kind of hill-descender, and assuming neighboring points in f differ in their y-value by at most 1, and assuming that every point except those at the bottom and top of the "bowl" have higher and lower neighbors, then if we run both algorithms for, say, 51 steps, then Φ will produce values of 1 for the histograms of a and it will produce values of 0 for the histograms of a'.

This shows that it is not unreasonable to believe that one algorithm could outperform the other algorithm on all N training runs. Thus, there are situations in which the unanimous choosing procedure will actually make a choice, and our previous claims are not merely vacuously true.

4.2.10 A Caveat

One possible caveat in showing that it is rational to believe that the unanimous choosing procedure is better than random is that our chosen extremely small probability $\delta = 2^{-\sigma} = 2^{-128}$ may not be small enough in certain situations. For example, if we use choosing procedures to pick between algorithms 2^{128} times, then it is very likely that during one of these uses a choosing procedure was fooled. This means that a probability of 2^{-128} will not be sufficient. Thus, the rate at which choosing procedures are used as compared to our chosen extremely small probability has a very important effect on whether this probability is "small enough."

Note however that we left open the possibility that the σ we use could be larger than 128. In fact, increasing σ by 1 means the rate at which choosing procedures are run would have to double in order for $2^{-\sigma}$ to again not be small enough. As it is, it is probably safe to say that choosing procedures are not run at

such a high rate that we should worry about our probability not being sufficiently small. If this were a concern though we could increase σ to a large enough number such that it is no longer a concern. Also, this potential problem is not unique to running choosing procedures. The same problem would occur in cryptography if decryption algorithms were run at an extremely high rate. Decryption algorithms are almost certainly run more often than choosing procedures, so if it is found that 128 bits of security is no longer sufficient in cryptography, then we can simply change our σ to an appropriately large value (that is sufficient for cryptographic security) as necessary.

4.3 Comparison to Work on the St. Petersburg Paradox

The apparent paradox that the results of this paper make is not a novel discovery. In fact, there is a similar paradox between mathematical probabilities and rational beliefs in a centuries old problem called the St. Petersburg paradox.

The St. Petersburg paradox comes about from a gambling game in which a fair coin is flipped until it comes up tails (see [8]). If a tail comes up on the first flip, there is a payout of \$2. If a tail does not come up until the second flip, then there is a payout \$4. And in general if a tail does not come up until the kth flip, then there is a payout of $\$2^k$. Thus, it is easily shown (see, e.g., [5]) that the expected payout of the game is arbitrarily large. Note that one possible reason this paradox occurs is because extremely low probability events, such as flipping a coin 128 times before a tail comes up, are used in calculating the expected payout.

With this arbitrarily large (in fact, infinite) expected payout, the question is then, how much should someone be willing to pay to play this game? According to [9], few people would pay even \$25 to play this game. That is, even though standard

probability theory tells us that any finite amount of money is an appropriate amount to pay, most rational people would not even pay $25. Thus, standard probability theory does not always provide a good model of rational real-world behavior [5]. This is similar to our paradox in which standard probability theory tells us to infer from the NFL theorems for optimization that general-purpose choosing procedures do not exist; but if it is rational to believe that extraordinarily small probability events will not happen, then we can show that it is possible to "break out" of these negative results.

Chapter 5

Discussion

5.1 Conclusion

As we have already shown, standard cryptographic practice indicates that extremely low probability events, such as decrypting a document encrypted using AES-128 with a limited number of guesses, are thought to have such an extraordinarily small chance of occurring that it is rational to believe that they will not occur. If we believe that this is true, then we can show that there exists (at least) one choosing procedure—the unanimous choosing procedure—that, on average, is better than random choosing and hence is a general-purpose choosing procedure. This is despite Theorem 2 which shows that, on average, the performance of any two choosing procedures is mathematically equivalent. This leads us to the conclusion that despite the seemingly negative results of our NFL theorem, the implications are not as negative as expected.

5.2 Future Work

One of the limitations in showing that the unanimous choosing procedure is superior to random guessing is that during the training runs one of the algorithms

always had to *win* in order for it to not randomly choose between the algorithms. That is, even if an algorithm tied sometimes, but still never lost, this algorithm was not chosen and instead an algorithm was randomly chosen. The reason for this is that once ties are allowed, the analysis becomes more complex. That is, we probably would not want to choose an algorithm if it won only once and then tied every other time; however, we might want to choose it if it tied once and won every other time. Thus some criteria for determining an appropriate cut-off for an allowable percentage of ties should be investigated.

Another, somewhat related, limitation is that in order to find a superior choosing procedure one of the algorithms *always* had to win. That is, the superior choosing procedure had to be the unanimous choosing procedure. In the future, some analysis of the implications of not requiring one algorithm to always win should be explored. For example, should we trust that our choosing procedure is better if it wins on 75% of the training runs? What about only 51%? Answers to these questions will give us a better understanding of when we should trust that a choosing procedure is correct in its decision.

Finally, similar to the NFL theorems for optimization, there are also NFL theorems for machine learning which show that any two learning algorithms are statistically equivalent (see [10]). Also, in [5], arguments analogous to those used in this paper show that there exist several general-purpose learning algorithms. Thus, the implications of the NFL theorems for machine learning are also not as negative as expected. It may be beneficial to combine these analyses.

References

[1] E. H. L. Aarts and J. K. Lenstra. *Local Search in Combinatorial Optimization*. Wiley-Interscience Series in Discrete Mathematics and Optimization. John Wiley and Sons, Ltd., Chichester, England (UK), 1997.

[2] D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997.

[3] D. L. Smitley and I. Lee. Comparative analysis of hill climbing mapping algorithms. Technical report, University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-88-94, November 1988.

[4] D. Angluin. Queries and concept learning. *Machine Learning*, 2:319–342, 1988.

[5] J. C. Jackson and C. Tamon. On the rationality of belief in free lunches in learning. Unpublished manuscript-in-preparation.

[6] National Institute of Standards and Technology. *FIPS PUB 197: Advanced Encryption Standard (AES)*. National Institute for Standards and Technology, Gaithersburg, MD, November 2001.

[7] Committee on National Security Systems. Fact sheet no. 1 for the national policy on the use of the advanced encryption standard (aes) to protect national security systems and national security information. Technical report, June 2003.

[8] D. Bernoulli. Exposition of a new theory on the measurement of risk. *Econometrica*, 22(1):23–36, 1954.

[9] I. Hacking. Strange expectations. *Philosophy of Science*, 47(4):562–567, 1980.

[10] D. H. Wolpert. The lack of a priori distinctions between learning algorithms. *Neural Computation*, 8(7):1341–1390, 1996.