Journal of Machine Learning Research 17 (2016) 1-42 Submitted 12/13; Revised 10/15; Published 4/16

Learning Using Anti-Training with Sacrificial Data

Michael L. Valenzuela MVALENZ@ECE.ARIZONA.EDU
Jerzy W. Rozenblit JR@ECE.ARIZONA.EDU
Electrical and Computer Engineering Department
University of Arizona
1230 E. Speedway Blvd.
Tucson, AZ 85721, UNITED STATES

Editor: Martin Pelikan

Abstract

Traditionally the machine-learning community has viewed the No Free Lunch (NFL) theorems for search and optimization as a limitation. We review, analyze, and unify the NFL theorem with the perspectives of "blind" search and meta-learning to arrive at necessary conditions for improving black-box optimization. We survey meta-learning literature to determine when and how meta-learning can benefit machine learning. Then, we generalize meta-learning in the context of the NFL theorems, to arrive at a novel technique called *anti-training with sacrificial data* (ATSD). Our technique applies at the meta level to arrive at domain specific algorithms. We also show how to generate sacrificial data. An extensive case study is presented along with simulated annealing results to demonstrate the efficacy of the ATSD method.

Keywords: Machine Learning, Optimization, Meta Optimization, No Free Lunch, Anti-Training, Sacrificial Data

1. Introduction

Most types of machine learning involve fitting a model to data. These models may be artificial neural networks, Bayesian networks, cluster centers, decision trees, etc. The fitting process uses objective/fitness functions to determine how well the learned model represents the training data. While many models (*e.g.,* artificial neural networks) have training algorithms (*e.g.,* backpropagation), it is not always clear which algorithm is best to use for a particular data set. This is the issue John Rice describes as the algorithm selection problem (Rice, 1976).

Knowledge about a problem must be used to solve it efficiently (Wolpert and Macready, 1997). It is a poor choice to use an algorithm that ignores derivatives or convex properties when the optimization problem exhibits these features. Similarly, it is a poor choice to use such features when they are absent. It may even be that the features change as the problem's parameters change. For instance, the number of local minima for a single neuron grows exponentially in the dimensionality of the input (Auer et al., 1995). The standard learning algorithm may not remain the most efficient as the dimensionality grows. Worse yet, some learning depends upon black-box optimizers such as simulated annealing (Goodsell and Olson, 1990), genetic algorithms (Wang et al., 2001), particle swarm (Zhang et al., 2004), and random search (Bergstra and Bengio, 2012).

The same virtue that makes black-box optimizers so widely used is also their inherent weakness—black-box optimizers use only a history of inputs into, and outputs from an objective function. This allows black-box optimizers to function on symbolic representations, experimental data, or results from simulations. The black-box optimizer does not use any knowledge about the symbolic function, data, or simulation. The No Free Lunch (NFL) theorems for search and optimization state that any such black-box optimization is expected to perform on average as well as a random guesser (Wolpert and Macready, 1997). If the objective function comes from a slow simulation or a physical experiment, then every function evaluation is precious. Thus, an algorithm tailored toward a particular problem distribution can make better use of the function evaluations.

In this work, we introduce a completely novel concept: *anti-training with sacrificial data*, for tailoring learning and optimization algorithms to problem distributions. At its base, anti-training is a generalization of a type of meta-learning. We take advantage of the fact that all optimization algorithms perform the same on average (according to the NFL theorems). Essentially, anti-training worsens the performance over problems suspected to be impossible or exceedingly unlikely to occur. Since the average performance remains constant, the performance elsewhere must increase. Consider the following water balloon analogy: the NFL theorems function as the conservation of volume. Squeezing the water (performance) out of one area of the balloon will transfer it to other areas.

Unlike many forms of meta-learning, anti-training inherently adheres to the NFL theorems since it is derived through the manipulation of the NFL theorems. Anti-training is expected to benefit learning performance whenever the objective functions are compressible. Moreover, because the sacrificial data used by anti-training is inexpensive to generate, it is possible to use anti-training where meta-learning cannot be applied. It can also be applied in conjunction with meta-learning and can prevent overtraining.

Following preliminaries and nomenclature in Section 2, we rewrite an NFL theorem to derive our technique and proceed to analyze it in Section 3. Section 4 discusses how to generate the sacrificial data required for anti-training. We extensively discuss several experiments in Section 5 and the results in Section 6. We then conclude with a discussion and future work in Section 7.

2. Preliminaries and Background

The original NFL theorems apply to arbitrarily large finite domains (*e.g.,* combinatorial optimization problems). Wolpert and Macready (1997) argue that this makes sense since computers only have a finite number of bits; the search space $\mathcal{X}$ and output space $\mathcal{Y}$ are typically bit sequences. Continuing with their notation, let f be a function representing an optimization problem (*e.g.,* error function, distance, etc.) and $\mathcal{F}$ be the set of all functions having an input $x \in \mathcal{X}$ and an output $y \in \mathcal{Y}$. If $|\mathcal{X}|$ is the size of the search space and $|\mathcal{Y}|$ is the size of the output space, then the size of the function space is $|\mathcal{F}| = |\mathcal{Y}|^{|\mathcal{X}|}$ (Wolpert and Macready, 1997). (Just to realize the order of magnitude of $|\mathcal{F}|$, consider a function with a 16-bit input and an 8-bit output; then $|\mathcal{F}| = 2^{2^{19}} \approx 2.6 \cdot 10^{157826}$. This is a large number of possible functions, but it is still finite.)

Moreover, let

- $f \in \mathcal{F}$ be a function to be optimized,

- $P(f)$ be a probability mass function, describing the probability of f occurring,
- $\hat{P}(f)$ be an estimate of $P(f)$,
- a be a black-box algorithm that samples f and uses the input-output data to generate the next input into f,
- $m \in \{1, \ldots, |\mathcal{X}|\}$ be the number of unique inputs,
- d_m^x be a set of the first m unique inputs,[1]
- d_m^y be a set of the corresponding outputs,
- d_m be shorthand for d_m^y and d_m^x combined,
- $P\left(d_m^y|f, m, a\right)$ be the conditional probability of finding d_m^y given an algorithm a iterated m times on a fitness function f, and
- $\Phi\left(d_m^y\right)$ be a function that converts the output to some performance metric.

For brevity, we provide the following terms:

- *anti-training with sacrificial data* is shortened to *ATSD*,
- *anti-training with sacrificial data combined with meta-learning* is abbreviated as *ATSD+ML*,
- *meta-learning* is abbreviated as *ML*, and
- *No Free Lunch* to be abbreviated as *NFL*.

2.1 NFL

Briefly, the NFL theorems state that for both static and time-varying optimization problems, all algorithms perform the same when averaged across all problems (Wolpert and Macready, 1997). We will focus on static problems. The NFL theorems are cast in a probabilistic framework:

Theorem 1

$$\sum_{f \in \mathcal{F}} P\left(d_m^y|f, m, a_1\right) = \sum_{f \in \mathcal{F}} P\left(d_m^y|f, m, a_2\right). \tag{1}$$

In the words of the authors: "this means in particular that if some algorithm's performance is superior to that of another algorithm over some set of optimization problems, then the reverse must be true over the set of all other optimization problems." The above equation is true regardless of the problem distribution, $P(f)$. This is different from saying all algorithms are equal. For all algorithms to be equal we require:

1. Algorithms are considered to never reevaluate a previously investigated input (allow previous function evaluations to be recalled without incrementing m).

Theorem 2

$$\sum_{f\in\mathcal{F}} P\left(f\right) P\left(d_m^y|f,m,a_1\right) = \sum_{f\in\mathcal{F}} P\left(f\right) P\left(d_m^y|f,m,a_2\right). \tag{2}$$

This explicitly depends on $P(f)$. Following the proof provided in Wolpert and Macready (1997), (2) requires

$$\sum_{f\in\mathcal{F}} P\left(f\right) P\left(d_1^y|f,m=1,a\right) = \sum_{f\in\mathcal{F}} P\left(f\right) \delta\left(d_1^y, f\left(d_1^x\right)\right) \tag{3}$$

to be independent of d_1^x and hence a. This sum is used in the base case for the inductive proof in Wolpert and Macready (1997), where for information theoretic reasons, they assume a $P(f)$ to be uniform. $P(f)$ could also be any distribution following $\prod_{x\in\mathcal{X}} P(y=f(x))$, or certain distributions with specific correlations between costs and inputs (Wolpert and Macready, 1997).

The NFL theorems have been extended to include additional priors and results. Schumacher et al. (2001) extend this to include sets of functions closed under permutations of the input-space. Thus, all algorithms sum to the same performance and generate the same collection of d_m when all functions are considered (Schumacher et al., 2001). Other analyses and extensions can be found in Culberson (1998); Droste et al. (2002); Corne and Knowles (2003a,b); Igel and Toussaint (2003, 2004); Giraud-Carrier and Provost (2005); Whitley and Watson (2005); Wolpert and Macready (2005); Auger and Teytaud (2007, 2010)). The results of these studies show that the No Free Lunch theorems are valid, but their relevance varies depending on the situation. Yet, despite all the extensions and debate, “there are no specific methods or algorithms that directly follow from NFL” (Whitley and Watson, 2005). In the paper, we provide a method that stems from the NFL theorems.

The NFL theorems treat the definition of “algorithms” differently than the typical definition of algorithm implies. In the NFL theorems, the “algorithms” are more about exploring the search space (points) in some order than about the actual instructions carried out by the computer. Two “algorithms” are considered identical if they always visit points in the same order, even if the underlying programming code is different. Likewise, the NFL theorems call two “algorithms” different even if they share the same code, but run with different hyper-parameters (*e.g.,* step-size, branching preference, etc.) so long as points are visited in a different order.

In the following subsections, we briefly summarize information theory and meta-learning prerequisites needed to establish the foundations for ATSD.

2.2 Information Theory

Information theory is a useful tool for studying learning (Bialek et al., 2001; Rissanen, 1984, 1986, 1987, 1989) and analyzing the NFL theory (Schumacher et al., 2001). In information theory, there are two common measures of complexity: “entropy” and “Kolmogorov complexity.” Entropy is typically defined as:

$$H(X) = \sum_{x\in\mathcal{X}} -P(x)\log P(x),$$

where X is a discrete random variable and $\mathcal{X}$ is the support of X. Entropy is sometimes interpreted as random variable's uncertainty. Kolmogorov complexity, sometimes called algorithmic entropy or complexity under analysis, cannot be specified so concisely. Essentially Kolmogorov complexity describes the size of the smallest program needed to reproduce a specified sequence (*e.g.,* bits, characters, numbers, etc.). Since there are multiple languages and machine architectures, Kolmogorov complexity is an incomputable function (Cover and Thomas, 2006; Li and Vitányi, 2008). In this paper, when we say something is compressible or incompressible, we mean this in the sense of Kolmogorov complexity. Information theory shows most functions are incompressible (a proof can be found in the work of Cover and Thomas (2006)). One should be careful to distinguish the ideas of a function being compressible versus being represented in polynomial space. If the number of bits needed to represent a sequence can be halved, there are still an exponential number of such bit sequences.

Culberson (1998) indirectly uses entropy when discussing the NFL theorems from different degrees of "blindness." The degrees of "blindness" range from complete information (the objective function is known) to no information. When these degrees of "blindness" are recast in a probabilistic framework, one can see that the random variable describing the objective function has zero entropy in the case of complete information and maximal entropy when nothing is known. Thus a main point of Culberson (1998) is that optimization performance degrades as the problem distribution becomes more complex.

2.3 Meta-learning

Before we begin our formulation of ATSD, due to the strong ties between ML and ATSD, it is important to address questions pertaining to the NFL theorems and ML. The first and probably most important question is: "can ML successfully improve performance in light of the NFL theorems?" If so, when does ML help? How can one judge the quality of an algorithm's learning biases?

ML is generally concerned with learning to learn better. Nevertheless, it can be applied to learning to optimize better. One perspective of ML is the algorithm selection problem (Rice, 1976). That is, given a specific instance of a problem, which algorithm solves the problem best? Giraud-Carrier and Provost (2005) address whether ML can escape the limitations imposed by the NFL theorems. Their ultimate conclusion is that only techniques that learn about the problem distribution $P(f)$ can overcome the NFL theorems. Hence, ML can be useful in light of the NFL theorems.

Vilalta and Drissi (2002) introduce two important ML concepts. One is that of structured versus random problems. Structured problems have lower Kolmogorov complexity than random problems. This structure can be exploited by algorithms to solve these problems more efficiently. By this definition, random problems have no such structure that ML can exploit. Consequently, ML is limited (with respect to its effectiveness) to structured problems. In terms of entropy, structured problems may still require exponential space to represent. To see this let there be $|\mathcal{F}|$ functions represented by 2^n bits, then there are about $2^{\alpha n}$ compressible functions with a Kolmogorov complexity of $2^{\alpha n}$, for $\alpha < 1$. The more structured the problem domain, the more quickly the percentage of structured problems approaches zero. Therefore, the percentage of structured problems decays geometrically with the size of $|\mathcal{F}|$. In other words, ML applies as long as most problems are considered irrelevant.

One may like to know how to measure one's success in selecting an algorithm. To do so, we first need a definition of success. Vilalta and Drissi (2002) introduce the concept of algorithm biases expressed as restrictions and rankings of potential hypotheses (the true f). In our formal notation, this means each algorithm assumes some $P(f)$. Then successfully selecting an algorithm means that the selected algorithm's assumed problem distribution matches the actual problem distribution. Both a theoretical and realistic measure exist to judge how well these match. Wolpert and Macready (1997) propose using the inner product between $P(f)$ and $P(d^y_m|f,m,a)$ over $f \in \mathcal{F}$ for a desired d^y_m. However, this method is only theoretical as computing the inner product is difficult due to the size of $\mathcal{F}$. A much more realistic measure is the off-training-set error (Wolpert, 2001).

We can use the answers to these three questions to lead us down a path of investigation. When ML samples from the problem distribution, the distributions are compressible (most problems are irrelevant), and the current method is suboptimal, then ML may operate within the NFL framework. Therefore, we will investigate how ML may manifest itself in the NFL framework. Second, due to the similarities between ML and ATSD, we will derive ATSD by applying an opposite goal to the opposite data. This in effect forms a "dual problem," which leads to our formulation of ATSD.

3. Anti-training

Section 2.3 gives us a research direction: derive ML from the NFL theorems, and subsequently, ATSD from that. To ensure our technique follows from (1), we begin by rewriting one side. First, we split $\mathcal{F}$ into three partitions (non-overlapping sets that cover the whole set):

$$\sum_{f\in\mathcal{F}} P\left(d^y_m|f,m,a\right) = \sum_{f\in\mathcal{F}_+} P\left(d^y_m|f,m,a\right) + \sum_{f\in\mathcal{F}_0} P\left(d^y_m|f,m,a\right) + \sum_{f\in\mathcal{F}_-} P\left(d^y_m|f,m,a\right). \tag{4}$$

Here $\mathcal{F}_+$, is the partition of problems already encountered, considered likely, or relevant; $\mathcal{F}_-$ represents the partition of problems that cannot occur (in the given context) or are considered unlikely to occur; and $\mathcal{F}_0$ consists of problems that are borderline likely or undefined. Equation 4 explicitly shows that we can trade-off performance between problem sets. Performance gained over $f \in \mathcal{F}_+$ must be lost over $f \in \mathcal{F}_0 \cup \mathcal{F}_-$.

The inquisitive reader may wonder why we use three partitions and particularly why we include the $\mathcal{F}_0$ partition. We include $\mathcal{F}_0$ for three reasons: first, an objective function does not always cleanly lie in either $\mathcal{F}_+$ or $\mathcal{F}_-$. This is because we avoid strict mathematical definitions of $\mathcal{F}_+$, $\mathcal{F}_0$, and $\mathcal{F}_-$. One may choose, for example, $\mathcal{F}_+ = \{f|P(f) > {}^{100}/_{|\mathcal{F}|}\}$ and $\mathcal{F}_- = \{f|P(f) < {}^{0.01}/_{|\mathcal{F}|}\}$. This flexibility necessitates $\mathcal{F}_0$. Second, $\mathcal{F}_0$ is useful for describing an important difference between ML and ATSD: the difference between the forthcoming (6) and (7). Third, when approximating $\mathcal{F}_+$ and $\mathcal{F}_-$, $\mathcal{F}_0$ is a catch-all for the remaining cases. Thus, three partitions are the minimum number needed to capture these degrees of freedom.

Equation 1 implies (4) is independent of a. However, each sum over a partition of $\mathcal{F}$ is generally allowed to change provided one important condition: $\mathcal{F}_+$, $\mathcal{F}_0$, and $\mathcal{F}_-$ are sets of functions not closed under permutation.[2] Schumacher et al. (2001) show in their Lemmas 1 and 2 that the NFL theorems apply if and only if the set of functions is closed under permutation. Since the fraction of sets closed under permutation is exceedingly rare (Igel and Toussaint, 2003), the NFL theorems fail to apply, with high probability, to each partitioning of $\mathcal{F}$. In turn, this means that each sum is generally allowed to change, implying that it is possible to improve the performance over $\mathcal{F}_+$.

Let us extend (4) in two ways. First, multiply by $\Phi\left(d_m^y\right)$ (a performance metric such as final error) and bring it inside the sum. Second, sum several $\sum_{f\in\mathcal{F}}\Phi\left(d_m^y\right)P\left(d_m^y|f,m,a\right)$ together for different values of d_m^y. Then we have

$$\sum_{d_m^y}\sum_{f\in\mathcal{F}}u=\sum_{d_m^y}\left(\sum_{f\in\mathcal{F}_+}u+\sum_{f\in\mathcal{F}_0}u+\sum_{f\in\mathcal{F}_-}u\right), \tag{5}$$

where

$$u=\Phi\left(d_m^y\right)P\left(d_m^y|f,m,a\right).$$

3.1 Anti-training as Meta-learning

Recall that a type of ML taking into account the NFL theorems is the one that attempts to learn $P(f)$ directly (Giraud-Carrier and Provost, 2005). However, this is often infeasible due to the cardinality of $\mathcal{F}$. A simple alternative to approximate this method is to learn an optimizer that performs well on the empirically common functions. This is a meta-optimization problem to improve the performance over $\mathcal{F}_+$. More accurately, the meta-optimization will improve the conditional expected value of $\Phi\left(d_m^y\right)$. Without loss of generality, hence forth assume Φ is a performance metric to be minimized (*e.g.,* an error function). Then, improving (minimizing) the conditional expected value means:

$$a^\star=\arg\min_a\sum_{d_m^y}\sum_{f\in\mathcal{F}_+}\Phi\left(d_m^y\right)\hat{P}_+(f)P\left(d_m^y|f,m,a\right).$$

However, the above equation is neither governed by nor does it follow from the NFL theorems! The introduction of a function dependent on f, in this case the approximation $\hat{P}_+(f)$, generally voids the NFL requirement that (3) is independent of a. Nevertheless, if we consider the typical approximation of $\hat{P}_+(f)$ used in practice, we realize that rarely does an identical objective function occur twice. See Appendix A for further justification of this assertion. This means $\hat{P}_+(f)$ is practically going to be a two-valued function: 0 and $\left|\hat{P}_+(f)\right|^{-1}$. Thus, in practice we drop the term $\hat{P}_+(f)$. This recovers the first term on the right-hand-side of (5), which follows from the NFL theorems:

$$a^\star=\arg\min_a\sum_{d_m^y}\sum_{f\in\mathcal{F}_+}\Phi\left(d_m^y\right)P\left(d_m^y|f,m,a\right). \tag{6}$$

2. A set of functions being closed under permutation means no new functions may be generated when the input space is bijectively remapped into itself (*i.e.,* a random permutation of the input space).

We make several more notes about the above equation. First, when using a deterministic a (or a random algorithm with a fixed random seed) $P\left(d_m^y|f, m, a\right)$ will be either 0 or 1; only a single d_m^y will be produced. This makes it easier to optimize in practice. Second, $\Phi\left(d_m^y\right)$ determines the behavior of $a^\star$; $a^\star$ could produce a low final error, low cumulative error, etc. While it is also possible to modify (6) to optimize over the cross-validation error, care must be taken as cross-validation alone is subject to the NFL theorem (Wolpert, 2001). Typically $\mathcal{F}_+$ will be approximated using samples from a problem distribution $P\left(f\right)$ arising from a limited domain. Additionally, (6) says nothing about the search for $a^\star$. Unless a is being modified intelligently, the meta-optimization problem will be subject to the NFL theorems itself. This means that finding better algorithms will be possible, but costly.

The NFL theorems say that any performance gained over a set of functions must be lost over the set's complement. Equation 4 must remain constant. Thus, ML improves the performance over $\mathcal{F}_+$ at the cost of performance over $\mathcal{F}_0 \cup \mathcal{F}_-$. This is potentially detrimental for prediction, generalization, and avoiding overfitting. To help mitigate this issue we now introduce *anti-training with sacrificial data* (ATSD). This means ATSD is about the optimization of one of the terms from (5):

$$a^\star = \arg\max_a \sum_{d_m^y} \sum_{f\in\mathcal{F}_-} \Phi\left(d_m^y\right) P\left(d_m^y|f, m, a\right). \tag{7}$$

This translates into

$$a^\star = \arg\min_a \sum_{d_m^y} \sum_{f\in\mathcal{F}_0\cup\mathcal{F}_+} \Phi\left(d_m^y\right) P\left(d_m^y|f, m, a\right).$$

Again, without loss of generality, we treat Φ as an error function to be minimized. As one can see, ATSD is an approximate dual of ML. ATSD achieves an improvement in the *sum* of performance over $\mathcal{F}_+ \cup \mathcal{F}_0$ by directly worsening the performance over $\mathcal{F}_-$. If the performance over $\mathcal{F}_-$ decreases, then by (4), the sum over $\mathcal{F}_0 \cup \mathcal{F}_+$ must improve. Despite ATSD being less efficient than ML, it has other benefits. For a more complete discussion, please see Sections 3.2 and 3.3.

It is critical to note that the performance improvement is over $\mathcal{F}_0 \cup \mathcal{F}_+$. Without additional care, ATSD could drop the performance over $\mathcal{F}_+$, but improve the performance over $\mathcal{F}_0$. To help avoid this, ATSD should be combined with ML (ATSD+ML). Combining (6) and (7), we have

$$a^\star = \arg\min_a \sum_{d_m^y} \left(\sum_{f\in\mathcal{F}_+} \Phi\left(d_m^y\right) P\left(d_m^y|f, m, a\right) - \beta \sum_{f\in\mathcal{F}_-} \Phi\left(d_m^y\right) P\left(d_m^y|f, m, a\right) \right) \tag{8}$$

where $\beta \geq 0$ determines the relative strength of anti-training. The above optimization technique is just one straightforward approach to combining ML and ATSD. ATSD+ML can be achieved by using a meta-optimization technique similar to the aforementioned process for ML. According to Bergstra and Bengio (2012), a random search for $a^\star$ is in some sense optimal. Although, preliminary tests suggest that local searches find locally optimal continuous hyper-parameters on continuous domains.

We feel it is important to note that ATSD+ML is not strictly limited to operating over $\sum_{f\in\mathcal{F}_+}$ and $\sum_{f\in\mathcal{F}_-}$. Other statistics can be used, too. For instance, the mean reduces to simply using (8) with an adjusted β. The median is another statistic that could be used. Meta-optimizing using the medians would be the same as dynamically selecting one or two f to improve or worsen their performance. For the same reason, percentiles are valid statistics to improve or degrade. Since percentiles could be used, one may wonder if the interquartile-range (IQR) is a valid statistic for ATSD+ML. ATSD+ML should avoid using the IQR statistic over $\mathcal{F}_-$, as this amounts to improving the performance over some $f \in \mathcal{F}_-$.

Algorithm 1 demonstrates the basic approach. The parameter *metaSearch* is a search algorithm for finding black-box search/optimization/learning algorithms; *metaMetric* is

Algorithm 1 Anti-Training Combined with Meta-Learning

```
 1: Procedure Anti-Meta-Search(
              metaSearch, metaMetric, pMetric, $\hat{\mathcal{F}}_+$, $\hat{\mathcal{F}}_-$, m)
 2: $metaOut \leftarrow empty$
 3: repeat
 4:   $searchAlg \leftarrow metaSearch(metaOut)$
 5:   $listPosPerfs \leftarrow empty$
 6:   for all $objFunc \in \hat{\mathcal{F}}_+$ do
 7:     $optResults \leftarrow$ Optimize($searchAlg$, $pMetric$, $objFunc$, $m$)
 8:     Append $optResults \rightarrow listPosPerfs$ {Accumulate results from $\hat{\mathcal{F}}_+$}
 9:   end for
10:   $listNegPerfs \leftarrow empty$
11:   for all $objFunc \in \hat{\mathcal{F}}_-$ do
12:     $optResults \leftarrow$ Optimize($searchAlg$, $pMetric$, $objFunc$, $m$)
13:     Append $optResults \rightarrow listNegPerfs$ {Accumulate results from $\hat{\mathcal{F}}_-$}
14:   end for
15:   $algPerformance \leftarrow$ metaMetric $(listPosPerfs, listNegPerfs)$ {e.g., (8)}
16:   Append $(searchAlg, algPerformance) \rightarrow metaOut$
17: until $algPerformance$ is good enough
18: Return Argmin($metaOut$)
```

```
 1: Procedure Optimize($searchAlg$, $performanceMetric$, $objFunc$, $m$)
 2: $d_m \leftarrow empty$
 3: for $itr = 1$ to $itr = m$ do
 4:   $x \leftarrow searchAlg(d_m)$
 5:   Append $(x) \rightarrow d_m^x$
 6:   $y \leftarrow objFunc(d_m^x)$
 7:   Append $(y) \rightarrow d_m^y$
 8:   Append $(d_m^x, d_m^y) \rightarrow d_m$
 9:   $performance \leftarrow performanceMetric(d_m)$
10: end for{Alternatively end when performance is good enough}
11: return $performance$ {If ending when performance is good enough return number of
    iterations}
```

the metric for preferring one black-box algorithm over another, such as (8); and *pMetric* is $\Phi\left(d_m^y\right)$, which rates the quality of outputs. The subroutine Optimize is a black-box search/optimization algorithm that runs the *searchAlg* to generate the input d_m^x into the black-box objective function *objFunc*.

Algorithm 1 starts by selecting, generating, or modifying a search/optimization/learning algorithm. For simplicity, let us assume the goal is optimization. The optimization subroutine generates the next input x from the history of inputs and outputs. The objective function (from $\hat{\mathcal{F}}_-$ or $\hat{\mathcal{F}}_+$) is evaluated at x. This process is repeated, building up the d_m vector. After enough iterations, the subroutine returns the performance metric, $\Phi\left(d_m^y\right)$. The performance over each $f \in \hat{\mathcal{F}}_+ \cup \hat{\mathcal{F}}_-$ is computed and stored using this subroutine. The meta-fitness function evaluates the optimization algorithm using ATSD+ML. This whole process is repeated for enough iterations or until the algorithm's performance is good enough.

3.2 Analysis

We now present how effective the ATSD is from a theoretical perspective. An extensive, empirical study follows in Section 5. It is difficult to analyze ATSD without making strong assumptions. Thus, we make the following assumptions as general as possible and defend them using logic similar to that used in the original NFL theorems (Wolpert and Macready, 1997).

Allow an algorithm to be represented by its learning biases as discussed by Vilalta and Drissi (2002). We choose to interpret these biases as an *a priori* rank ordering of functions. To help illustrate this point, imagine an algorithm represented by a deck of cards, where each card represents an objective function f to be optimized. Denote $R_a(f) \in \left\{1, 2, \ldots, |\mathcal{F}|\right\}$ as algorithm a's ranking (i.e., position in the deck) of f. A lower ranking means the algorithm is more biased toward finding the optimal solution in fewer iterations. The function with rank one will have its optimum input guessed first. Then all $f \in \mathcal{F}$ in violation of the first input-output pair, d_1, are removed from the deck of hypotheses and the process is repeated.

Allow $\hat{\mathcal{F}}_+$ and $\hat{\mathcal{F}}_-$ to be samples from $\mathcal{F}_+$ and $\mathcal{F}_-$ respectively. Furthermore, since we are working with domain specific optimizers, we may assume the majority of problems are not highly relevant:

$$\begin{aligned}\left|\mathcal{F}_+\right| &\ll \left|\mathcal{F}_0\right| \\ \left|\mathcal{F}_+\right| &\ll \left|\mathcal{F}_-\right|.\end{aligned}$$

Rather than debate the percent $f \in \mathcal{F}_-$, allow

$$|\mathcal{F}| = c\left|\mathcal{F}_-\right|,$$

for a constant c. For example, if $\left|\mathcal{F}_-\right| \approx \left|\mathcal{F}_0\right|$ then $c \approx 2.0$. Nevertheless, we will continue to use c symbolically to remain more flexible and general, allowing the reader to impose their own assumptions.

We will assume that the learning tasks are compressible, as this is necessary for meta-learning to work (Vilalta and Drissi, 2002). We will interpret compressibility as implying a correlation between rankings of functions. Particularly, if through meta-optimization some

$f \in \hat{\mathcal{F}}_+$ lowers in rank, then on average allow all $g \in \mathcal{F}_+$ to lower proportionally by a factor of ρ_+.[3] This will provide the average behavior for describing why ML works. To capture the corresponding effects of ATSD, when some $f' \in \hat{\mathcal{F}}_-$ rises in rank due to meta-optimization, allow all $g' \in \mathcal{F}_-$ to rise proportionally by a factor of ρ_-. We use the symbols g and g' to denote objective functions that are "dragged" along due to correlations in rankings. The two factors, ρ_+ and ρ_-, are functions of the compressibility of $\hat{\mathcal{F}}_+$ and $\hat{\mathcal{F}}_-$ respectively.

Assume that the meta-optimization task improves/degrades the rankings by p_c (percent change). That is, for $f \in \hat{\mathcal{F}}_+$ and $f \in \hat{\mathcal{F}}_-$ the ranking changes from $R_a(f)$ to

$$\begin{gathered} R_a\left(f\right)\,\left(1-p_c\right)+p_c, \text{ and} \\ \left|\mathcal{F}\right|\,p_c+R_a\left(f\right)\,\left(1-p_c\right) \end{gathered}$$

respectively. Thus the differences in rank after ML and subsequently ATSD are approximately

$$\begin{gathered} \big(R_a\left(f\right)-1\big)\;p_c, \text{ and} \\ \big(\left|\mathcal{F}\right|-R_a\left(f\right)\big)\;p_c. \end{gathered}$$

If we assume the starting algorithm is chosen at random, then we can assume the rankings are uniformly distributed: $P(R_a(f)) \sim U\big(1, |\mathcal{F}|\big)$. This follows from the same reasoning Wolpert and Macready (1997) use to argue that $P(f)$ ought to assume a uniform distribution when lacking further information. Then

$$E\left[R_a\left(f\right)\right]=\frac{\left|\mathcal{F}\right|+1}{2}$$

Now, we will proceed to show that the expected rank improvements from ML and ATSD are comparable. After accounting for rank correlations, improving a single $f \in \hat{\mathcal{F}}_+$ produces on average

$$\mu_+\left(f\right)=p_c\left(R_a\left(f\right)-1-\big(R_a(f)-1\big)\,\rho_+\sum_{g\in\mathcal{F}_+}\big(R_a(g)-1\big)\,\rho_+\right)$$

rank improvements. The term $\big(R_a(f)-1\big)\,\rho_+$ is subtracted to avoid double counting f as $f \in \mathcal{F}_+$, and the term $R_a(g)|g \in \mathcal{F}_+$ is any function that may have its rank improved due to correlations between problems. Substituting in the expected values of $R_a(f)$ and $R_a(g)$ produces

$$\begin{aligned} E\left[\mu_+\left(f\right)\right] &= p_c\left(\frac{\left|\mathcal{F}\right|-1}{2}\left(1-\rho_+\right)+\frac{\left|\mathcal{F}\right|-1}{2}\left|\mathcal{F}_+\right|\rho_+\right) \\ &= p_c\left(\frac{\left|\mathcal{F}\right|-1}{2}\right)\left(1+\Big(\left|\mathcal{F}_+\right|-1\Big)\,\rho_+\right). \end{aligned} \tag{9}$$

3. Since the effects of correlation are modeled in a deterministic fashion, this model can only be used to assess the expected values and not higher moments, such as variance.

Next we will compute the number of ranks degraded during ATSD. After accounting for correlations, but being careful to avoid double counting f', the number of ranks degraded on average for anti-training a single $f' \in \hat{\mathcal{F}}_-$ is

$$\mu_-\left(f'\right) = p_c\left(\left(|\mathcal{F}| - R_a\left(f'\right)\right)(1-\rho_-) + \sum_{g' \in \mathcal{F}_-}\left(|\mathcal{F}| - R_a\left(g'\right)\right)\rho_-\right).$$

Substituting in the expected values of $R_a(f')$ and $R_a(g')$ produces

$$E\left[\mu_-\left(f'\right)\right] = p_c\left(\frac{|\mathcal{F}|-1}{2}\right)\left(1 + \left(\left|\mathcal{F}_-\right| - 1\right)\rho_-\right). \tag{10}$$

We cannot directly compare (9) and (10) yet. Equation 10 is in terms of ranks degraded for $f' \in \mathcal{F}_-$, not ranks improved for $f \in \mathcal{F}_+$. We can calculate the expected number of $f \in \mathcal{F}_+$ promotions as a result of anti-training demotions by multiplying (10) by the ratio $\left|\mathcal{F}_+\right| / |\mathcal{F}|$. Recalling $|\mathcal{F}| = c\left|\mathcal{F}_-\right|$, we have

$$E\left[\mu'_-(f)\right] = p_c\left(\frac{|\mathcal{F}|-1}{2}\right)\left(\zeta + \left(\frac{\left|\mathcal{F}_+\right|}{c} - \zeta\right)\rho_-\right)$$

where $\zeta = \frac{|\mathcal{F}_+|}{c\left|\mathcal{F}_-\right|}$. Thus the relative power of ATSD versus ML is given by

$$\frac{E\left[\mu'_-(f)\right]}{E\left[\mu_+(f)\right]} = \frac{\left(\zeta + \left(\frac{|\mathcal{F}_+|}{c} - \zeta\right)\rho_-\right)}{\left(1 + \left(\left|\mathcal{F}_+\right| - 1\right)\rho_+\right)}. \tag{11}$$

Note $\zeta = \frac{|\mathcal{F}_+|}{|\mathcal{F}|} \approx 0$ by assumption. Provided a sufficiently large $\left|\mathcal{F}_+\right|$, we are justified in dropping additive terms on the order of 1 or less. Then (11) becomes

$$\frac{E\left[\mu'_-\right]}{E\left[\mu_+\right]} \approx \frac{\rho_-}{c\,\rho_+}. \tag{12}$$

If we allow ρ_- and ρ_+ to be on the same order of magnitude and let $c \approx 2.0$, then ATSD will be roughly half as powerful per training sample as ML.

3.3 Other Considerations

Here, we consider several questions about ATSD+ML. Can performance over some elements in $\mathcal{F}_-$ decrease, only to increase performance on other elements in $\mathcal{F}_-$? What happens if $\mathcal{F}_-$ is specified incorrectly? What benefits does ATSD+ML have over traditional ML?

Even though performance over the sacrificial data $\mathcal{F}_-$ is minimized, there is no reason to believe that all $f \in \mathcal{F}_-$ always perform worse after optimization. Yes, performance on some elements in $\mathcal{F}_-$ may increase. However, ATSD+ML only requires the *net* performance over $f \in \mathcal{F}_-$ to decrease. There are two things that can make the net performance over $\mathcal{F}_-$ improve during the meta-optimization procedure (8). First, $\mathcal{F}_-$ may be too "similar" to $\mathcal{F}_+$, meaning the algorithms being investigated treat both sets of problems similarly. This can occur when the search for $a^\star$ is incomplete, such as when searching only for an optimizer's hyper-parameters instead of program code. The fixed program code may be biased toward treating both sets similarly. Alternatively, the β value in (8) needs to be greater.

ATSD's behavior is similar to ML's behavior when the data sets are specified incorrectly. To the contrary, ATSD+ML is more robust than ML when $\hat{\mathcal{F}}_-$ is specified correctly. Consider the case when $\hat{\mathcal{F}}_+$ poorly approximates $\mathcal{F}_+$. If the approximation is sufficiently poor, performance over $\mathcal{F}_+$ will decrease and performance over $\mathcal{F}_-$ will change randomly. ATSD+ML helps ensure a drop in performance over $\mathcal{F}_-$, regardless. This is assuming $\hat{\mathcal{F}}_-$ is specified correctly, which is generally easy. Section 4 addresses this issue. Alternatively, consider the scenario where $\hat{\mathcal{F}}_+ = \hat{\mathcal{F}}_-$. One of three things may happen according to (8). If $\beta < 1$, then the behavior defaults back to traditional ML. If $\beta = 1$, all algorithms appear equal and nothing happens. If $\beta > 1$, then the performance $\hat{\mathcal{F}}_+$ worsens. In general, even when $\hat{\mathcal{F}}_+ \neq \hat{\mathcal{F}}_-$, a safe value for β is $\beta < \left|\hat{\mathcal{F}}_+\right| / \left|\hat{\mathcal{F}}_-\right|$, as this ensures the traditional ML term dominates the meta-optimization procedure. However, to strictly follow the NFL derivation one should set $\beta = \left|\hat{\mathcal{F}}_+\right| / \left|\hat{\mathcal{F}}_-\right|$ so that each objective function is weighted equally.

ATSD's use of $\mathcal{F}_-$ helps define where the learning algorithm will fail. Specifying some of the failure cases removes some unknown failure cases. This follows from the fact, as shown by Schumacher et al. (2001), that all algorithms produce the same collection of d_m^y when all functions are considered. Also, in theory when $\mathcal{F}_-$ includes a model of noise, then the algorithm learns to fail at fitting the noise. This is entirely different from using a less powerful model that cannot represent the noise. The algorithm could still represent the noise, but ATSD would make it more difficult for the algorithm to find models that do so.

ATSD also helps when data is hard to come by. When real problems or data are scarce or expensive, ML is limited. Just like normal data fitting, if few data points are provided then overfitting is problematic. However, $\mathcal{F}_-$ can be easy and cheap to generate (for more on this see Section 4). As long as there is a balance between preserving/maximizing performance over a small $\mathcal{F}_+$ and minimizing the performance over a large $\mathcal{F}_-$, then this should help avoid overfitting $\mathcal{F}_+$.

4. Generation of Sacrificial Data

In this section, we present five methods for generating sacrificial data (*i.e.,* $f \in \mathcal{F}_-$). We classify these methods into three categories: randomized, empirical, and theoretical. First, we discuss the possibility and consequences of generating the sacrificial data purely randomly. Yet randomly generated sacrificial data may still lie in $\mathcal{F}_+$, so next we discuss rejecting randomly generated sacrificial data that is too similar to $\hat{\mathcal{F}}_+$ (using confidence intervals or other measures of similarity). A similar approach starts with functions in $\mathcal{F}_+$ and "destroys" patterns that are initially present. A different approach is to use empirical "pure" noise. Last, we analyze the efficacy of using a model similar to the source for $f \in \mathcal{F}_+$, but changing the model to violate logical/physical laws, use unlikely distributions, etc. Since $\mathcal{F}_-$ will usually be large, $\hat{\mathcal{F}}_-$ may be generated with any combination of these techniques.

One technique is to randomly generate functions. Surprisingly, this method works with a high probability of success given a sufficiently large $|\mathcal{F}|$. Vilalta and Drissi (2002) conjecture that failing to learn unstructured tasks (*i.e.,* tasks with a large relative Kolmogorov complexity) carries no negative consequences. As discussed in Section 2.3, the percentage of structured problems decays geometrically with the size of $|\mathcal{F}|$. Thus, for a sufficiently large problem space, randomly generated objective functions will be, with high probability, unstructured.[4] Therefore, we can exploit structure present in learning tasks with high probability, even without knowing the structure. Ergo, sacrificial data can be randomly generated for problems with sufficiently large input and output spaces. While randomly generated functions may work with high probability, they may be a poor choice for the sacrificial data. Consider the result given by (12). Due to the nature of randomly generated functions, they will probably have larger Kolmogorov complexity and hence a smaller ρ_- than other methods.

If randomly generated sacrificial data is too similar to a problem in $\hat{\mathcal{F}}_+$, it may be rejected. This can be accomplished with confidence intervals or other measures of similarity. Suppose the objective function's output is suspected to follow an unknown ergodic process. Since ergodic processes have estimable statistical properties (by definition), confidence intervals can be constructed on these statistics. New random functions can be generated to have their output lie outside these confidence intervals. One could analyze other metrics besides the functions' outputs, such as correlations, covariance, and mutual information between the input and output. By appropriately selecting the confidence intervals, the randomly generated functions can be in $\mathcal{F}_-$ with arbitrary confidence. However, randomly generating functions will (probabilistically) have a small ρ_-.

To improve ρ_-, one approach would be to take problems from $\hat{\mathcal{F}}_+$ and modify the functions with a few simple steps. This will barely alter the Kolmogorov complexity, by its very definition (*i.e.,* minimally sized program to generate the data). Yet, this approach may be ineffective unless these modifications are "natural." Since the Kolmogorov complexity is dependent on the description language (Li and Vitányi, 2008) and the description language is dependent on the optimizer, the types of modifications should somehow match the possible optimizers. This is what we mean when we say "natural." For instance, consider an attempt to determine an optimal step-size for a local search over a set of smooth objective functions. Generating $\hat{\mathcal{F}}_-$ using simple pseudo-randoms would be "unnatural," and possibly result in a minuscule ρ_- in (12). Whereas, scaling the smooth functions may be more useful for generating $\hat{\mathcal{F}}_-$. Again, to ensure the modified functions belong to $\mathcal{F}_-$, the modifications should destroy any patterns originally present, modify correlations, and alter other metrics.

For objective functions that are data-based, another technique is to generate objective functions $f \in \mathcal{F}_-$ from signal-less empirical noise. Most would agree that a learner failing to see patterns in empirical noise is a positive quality. If it is expensive to collect large quantities of empirical noise, then pseudo-random noise could be generated from a random process modeling the empirical noise. If the empirical noise cannot be measured

4. Hardware random generators make it easier to generate functions with high Kolmogorov complexity.

Table 1: Experiment summary

Name	Objective type	Meta-iterations	Trials $\times$ subtrials	Tests	Hypothesis
Satellite Trajectory	Continuous optimization	≈ 900	4×4	100	ATSD works
High-dimensional TSP	Discrete optimization	10,000	5×4	10,000	ATSD fails
Low-dimensional TSP	Discrete optimization	2,000	4×200	400	ATSD works
$50k Classification	Learning with a SVM	3,500	10×1	1	ATSD works

directly, time-series analysis offers standard procedures for isolating noise from data carrying trends (Gershenfeld, 1999; Brockwell and Davis, 2002; Shumway and Stoffer, 2011).

The last procedure for generating $\hat{\mathcal{F}}_-$ makes use of models, simulations, or subject matter experts (SMEs). The models or simulations can be altered to contradict logical or physical laws. Replacing typical distributions with unlikely or impossible distributions can likewise lead to $f \in \hat{\mathcal{F}}_-$. Those $f \in \mathcal{F}_-$ that come from models and simulations will tend to have lower Kolmogorov complexity than randomly generated functions. We conjecture that this translates into a larger ρ_-. Even without a simulation, SMEs can help provide high quality $\mathcal{F}_-$ due to their knowledge of what is likely, unlikely, and impossible. Furthermore, models, simulations, and SMEs provide an implicit distribution of $P(f)$. Yet, the downside to this approach is that it takes a significant amount of knowledge about the domain from which the problems come, so it might be impractical. This approach is most useful when much is known about the problems, but little is known about which optimizer or learner to use.

5. Experiments

To help verify the theory and investigate the possible interactions between traditional ML and ATSD, we conducted four, very extensive experiments mimicking how meta-optimization procedures are used in practice. The first experiment focuses on finding an efficient algorithm for optimizing a satellite's trajectory to monitor space debris. The second experiment involves the search for an algorithm that performs well on highly random traveling salesperson problems (TSP). The third experiment is also a TSP, but this time the cities occur on a ring and their distances are calculated from that. The last experiment tries to maximize the predictive accuracy on a classification problem. We summarize these experiments, their objectives, size, and purpose in Table 1.

In Table 1 "objective type" shows what the meta-optimization process was optimizing. The column titled "meta-iterations" describes how many unique algorithms were tried in the search for the best algorithm for the task. "Trials $\times$ subtrials" describes how many ML and ATSD trials were conducted. Specifically, a trial refers to a specific configuration with a fixed $\hat{\mathcal{F}}_+$ and $\hat{\mathcal{F}}_-$. Subtrials were conducted to capture the variability of the meta-optimized algorithms found after the specified number of meta-iterations. Each resulting algorithm produced by the meta-optimization process was then tested across the number of tests listed in the "tests" column. Lastly, the column titled "hypothesis," describes our hypothesized outcomes according to theory.

Throughout the course of conducting the experiments, we discovered multiple possible ways the meta-optimization process may fail, for both ML and ATSD. When the search for optimizer algorithms or supervised learning algorithms is constrained (*e.g.,* only adjusting hyper-parameters), the algorithms may fail to exploit structure, which is embedded in the problem. This was demonstrated with the high-dimensional TSP; it has structure that the investigated optimizers have no capabilities to exploit. Another way for the process to fail is when the number of unique function evaluations is not fixed. Unless the number of unique function evaluations is fixed, some algorithms will use fewer unique function evaluations on problems from $\mathcal{F}_-$. The next problem is two fold: (1) the meta-optimization may fail to return a near optimal algorithm for the approximate problem distribution ($\hat{\mathcal{F}}_+$ and $\hat{\mathcal{F}}_-$), and (2) this approximation of the problem distribution may be poor. Thus, the accuracy of the approximation and the quality of the meta-optimization solution must be balanced;[5] the optimal algorithm for a poor approximation is undesirable. These problems affect both ML and ATSD alike.

Below we describe each experiment in more detail, including the domain where the objective functions come from, the meta-optimization procedure, which common ML practices we follow, individual tests within the experiment, and the computer setup. Results for each experiment are presented in Section 6. Just a quick note before delving into the first experiment. Whenever we say something is "low-level," we are referring to the base optimization or learning problem. Similarly, if we mention something is "meta-level" we are referring to the meta-optimization required for ML and ATSD that optimizes the "low-level" algorithm.

5.1 Satellite Trajectory Problem

For the first experiment we use a physics based domain where the pilot problems involve monitoring space debris around a small artificial planet. Monitoring space debris presents a trade-off between the quality of observations and the time until revisiting the same debris. If the satellite were to have nearly the same orbit as a single piece of debris, then the satellite would pass by the debris slower and closer, resulting in higher quality observations. Similarly, a different orbit that is closer to or further from the planet would result in seeing the debris more frequently, but from further away. The low-level optimization objective is to find an orbit for a satellite that visits all debris within a specified distance and with minimal mean-time between visits.

We simplify the domain for the experiment in the following ways. Gravity of the planet is assumed to be uniform. Space debris are treated as points, meaning they have no physical size. Collisions are ignored. We use only 10 pieces of space debris to minimize simulation time. All orbits are in a two-dimensional space. Thus, an orbit's degrees of freedom are its: eccentricity (shape), orientation (angle of the ellipse), semi-major axis (size), and true

5. One may liken this scenario to the processing inequality in information theory.

Table 2: Comparison of Pilot Problem Distributions

Data Set	$\mathcal{F}_+$	$\mathcal{F}_-$
Semi-Major Axis	$\sim N(400km, 30km)$	half from $\sim N(520km, 30km)$ half from $\sim N(280km, 30km)$
Eccentricity	$\sim N(0.1, 0.1)$	$\sim Exp(1/3)$, but ≤ 1.0
True Anomaly	$\sim U(0, 2\pi)$	$\sim U(\pi/4, 7\pi/4)$
Orientation	$\sim U(0, 2\pi)$	$\sim U(\pi/2, 2\pi)$

anomaly (phase of the orbit). In spite of these simplifications and the fact all the variables are continuous, the multiple pieces of space debris introduce many local extrema.

Recall that our methods discussed in Section 3.1 require multiple objective functions. We create multiple instantiations of problems from this domain. Each problem has space debris in a different configuration, but drawn from one of two distributions. These distributions correspond to $\mathcal{F}_+$ and $\mathcal{F}_-$. More details can be found in Table 2. To ensure there were no large gaps, we used Latin hypercube sampling (McKay et al., 1979).

5.1.1 Meta-Optimization Procedure

The meta-level problem that we investigate is the meta-optimization of an optimizer. We chose to use Matlab's simulated annealing (SA) procedure as our low-level optimizer. SA was chosen for this experiment due to its large number of hyper-parameters and behavior dependent upon these hyper-parameters (Ingber, 1993, 1996). The hyper-parameters we investigate are:

- step length function (two choices),
- initial temperatures (four real numbers),
- cooling function (three choices),
- reannealing time (one integer),
- upper bounds (four real numbers), and
- lower bounds (four real numbers).

The meta-optimization procedure is designed to find an optimizer that quickly minimizes the mean-time between the satellite visiting debris and the satellite's closest approach to each piece of debris. It does this by searching for good values for the above hyper-parameters. We searched for hyper-parameters that produce a low cumulative objective value, meaning finding a lower objective value sooner was preferred (*i.e.*, $\Phi\left(d_m^y\right) = \sum_{i=1}^m \min\{d_i^y\}$).

The meta-optimization consisted of three phases: differentiation, SA optimization, and a local search. The starting hyper-parameters were differentiated using SA for 35 iterations. While we recorded only the best solution during these 35 iterations, all remaining optimization iterations were recorded. Then, SA was used to further investigate the hyper-parameters, due to its global search properties.[6] Last, a gradient descent method (Matlab's

6. We acknowledge the NFL-theorems apply to meta-search.

Table 3: Four Types of Trials

Methods	ATSD+ML3	ML9	ML3	Control
$\lvert\hat{\mathcal{F}}_+\rvert$	3	9	3	1
$\lvert\hat{\mathcal{F}}_-\rvert$	9	0	0	0
Subtrials	4	4	1	1
Control?	No	No	Yes	Yes

`fmincon`) was used to finish the search process. We discovered that gradient descent worked well to search for real-valued hyper-parameters on this particular problem. The meta-optimization problem was setup to minimize the median of $\hat{\mathcal{F}}_+$ and maximize the median of $\hat{\mathcal{F}}_-$. We used $\beta = 1/3$, because our ATSD+ML test uses three times as many sacrificial functions as probable functions.

5.1.2 Practices Followed

One common practice involves searching for hyper-parameters (e.g., model selection) to minimize the bootstrap- or cross-validation error (Kohavi, 1995; Arlot and Celisse, 2010). Following these practices, we tested ATSD+ML according to (8), to search for good hyper-parameters for an optimization procedure on a collection of pilot problems. However, our experiment differs in three key aspects from common practice. First, we only test (i.e., judge the final performance) against data the optimizer has never seen, as stipulated by Wolpert (2001); Giraud-Carrier and Provost (2005). Second, the meta-optimization occurs across multiple objective functions from a limited domain, as advised by Giraud-Carrier and Provost (2005). Another common practice we purposefully chose to ignore is the partitioning of the data into training, probe/validation, and test partitions. We only use training and test partitions. The reason for this is the need to test if ATSD+ML limits the effects of overtraining. We test the algorithms from only the final iteration for the same reason.

5.1.3 Trials

We conduct four types of trials, summarized in Table 3: ATSD+ML3, ML with three samples (ML3), ML with nine samples (ML9), and a control where no ML nor ATSD are applied. ML9's $\hat{\mathcal{F}}_+$ consisted of nine sample problems. ML3 and ATSD+ML3 share a subset of ML9's $\hat{\mathcal{F}}_+$. The control uses one problem from the set ATSD+ML3 and ML3 share. We chose these trials to demonstrate how performance can be improved. The ML3 performance could be improved by either increasing the number of samples from $\mathcal{F}_+$ (ML9) or by introducing ATSD (ATSD+ML3). Both these alternatives are tested four times. Each algorithm generated from the table is then tested against 100 samples of never seen before data from the $\mathcal{F}_+$ distribution. We leave the analysis over $\mathcal{F}_0$ for another experiment.

5.1.4 Experiment Computer Setup

We ran the experiments using a cluster of three standard-grade (as of the year 2008) computers. One computer is a Gateway E6610Q model, using the Intel QX6700 quad-core processor. Two other computers are custom built, one using the Intel Q6600 quad-core processor and the third using the Intel i7-3770K quad-core processor. All computers have 4GB of RAM (DDR2 800, DDR3 1033, DDR3 1600). All file I/O is carried out using a virtual drive hosted over the Internet. Both results and computations that could be reused are saved to the virtual drive. Two of the three computers run Matlab on Windows 7. The third computer runs Matlab in Ubuntu.

We acknowledge that this hardware is now dated and that there are better suited environments that should, and will, be used in the future. However, due to the lengthy process to gain access to high performance computing clusters, we only use a small cluster here. Anyone wishing to reproduce the results or to extend them will easily be able to use the coarse-grain parallelism that we built into our code to exploit larger clusters.

The meta-optimization code is written to exploit coarse-grain parallelism, to take advantage of each core available, while avoiding the need to use Matlab's parallel computing toolbox. Each computer performs meta-optimization for one of the ATSD+ML3, ML9, and ML3 trials. To keep one computer responsive for daily tasks, only one trial was performed on ML3. The control case did not use any meta-optimization.

The meta-optimization process takes about 1.5 months to finish for the ML3 experiment, about 4.5 months to finish the ML9 experiment, and approximately 6.0 months to complete the ATSD+ML3 experiment. After six months, we collect about 1.7 GB of data.[7] After the meta-optimization, we run the algorithms over 100 different problems from the $\mathcal{F}_+$ to collect statistical performance data. Each algorithm takes about 10 hours to run over this set of 100 problems. Due to how relatively quick it is to test an algorithm over 100 problems, we did not need to partition the trials over the cluster. This generates another 1.4 GB of data over the course of a week. All 3.1 GB of the data is available, but we also offer just the Matlab scripts (only 100KB) that can generate statistically similar data.

Because the meta-optimization run times were deemed too slow to reproduce statistically similar results, we found ways to improve the execution speed. Since the core kernel of the simulation is already optimized and Matlab's ODE45 was still the bottleneck (taking over 98% of the time), we copied the ODE45 code and removed all the code for options. Further improvements were made by turning off Matlab's ODE45's interpolation, as we were already interpolating ODE45 results at specific points.

These changes give us approximately a 2x speedup in total. Trimming the ODE45 code improves the speed by about 50%. We are surprised how much this helps; we attribute it to avoiding the majority of the ODE45's if-branches, removing its switch statement, and reduction in code size. Turning off Matlab's ODE45's interpolation provides another 30% speed up. With modern hardware and these revisions, it should take only approximately two months to produce statistically similar results.

5.2 High Dimensional Traveling Salesperson Problems

In this experiment we try to demonstrate that theory correctly predicts when ATSD should fail. Theory predicts that if ML fails, then so should ATSD. We believe this is an important experiment as it tries to disprove the theory from the opposite direction; we would be

7. Most of the data was saved computation to avoid resimulating space debris trajectories and check-pointing to prevent loss of data in advent of system crashes. Unfortunately some of the earlier data was lost, as the check-pointing system was implemented after the first system crash.

Table 4: Correlation Matrix by Problem Distribution

	$\mathcal{F}_+$	$\mathcal{F}_0$	$\mathcal{F}_-$
$\Sigma_{i,j} =$	$\frac{2.82}{(1+\lvert i-j\rvert)^2}$	$\frac{0.5}{\lvert i-j\rvert^2}$	$\delta(i,j)$

surprised to find that ATSD works where ML fails. ML and ATSD require structure in the input-output pairs. Structure, only in the problem itself but not the input-output pairs, is of little use for an optimizer or learner unless one is attempting to discover the underlying problem (*e.g.,* the TSP cost matrix) and then solve it with a non-black-box solver. This is actually one method discussed in (Culberson, 1998).

For all TSP problems in this experiment, we limit the input to only valid sequences of cities. Thus, the entire input space is valid. Only total distance traveled is analyzed. No information from partial solutions is used. The problem distributions are generated with the following technique:

- For each row of the cost matrix, sample a 20 dimensional multivariate normal distribution.
- Compose a cost matrix from 20 such samples.
- Make the cost matrix symmetric by adding it to its own transpose.
- The minimum value is set to zero.

The covariance matrix is defined in Table 4. $\mathcal{F}_+$ has a nearly singular covariance matrix, producing 20 random numbers, which approximately follow a random walk.[8] Before making the cost matrix symmetric, each row is an independent sample. This structure is not readily exploitable by the low-level optimizer. The low-level optimizer avoids analyzing trends in these random walks. $\mathcal{F}_-$ produces a random cost matrix where each element is independently normally distributed. We will assume this is sufficient to make the probability of particular outputs independent of the inputs, implying $P(f) = \prod_x P(y = f(x))$.

5.2.1 Meta-Optimization Procedure

The meta-optimization problem is to find the hyper-parameters for a custom genetic algorithm. The hyper parameters include five initial paths, a mutation rate, a max swap-size schedule, and a swap-size parameter. The initial paths and mutation rate should be self-explanatory. The swap-size parameter defines a random variable distributed according to the exponential distribution, but this value is clamped to never exceed the max swap-size. The max swap-size schedule provides 10 maximum values, each one valid for 10% of the low-level optimization procedure. The clamped exponentially distributed swap-size determines how many cities are swapped at once.

The custom genetic algorithm primarily differs from a traditional genetic algorithm in that it only generates valid inputs for the TSP problems. Random mutations are implemented as random swaps. The crossover operation uses the differences in the paths

8. It is not technically a random walk, but when plotted the values resemble a random walk.

Table 5: Five Types of Trials

Methods	ML5	ML10	ML15	ATSD250+ML5	ATSD250+ML5 Weak
$\left\|\hat{\mathcal{F}}_+\right\|$	5	10	15	5	5
$\left\|\hat{\mathcal{F}}_-\right\|$	0	0	0	250	250
Subtrials	4	4	4	4	4
β	0	0	0	1	0.2

of the two top performing individuals to guide which cities to swap. We use a population size of five. At the time the experiment was run, the implementation permitted duplicate individuals. This means it is probable that the number of unique function evaluations differ for all trials; this is significant since any subtrial may be put at a disadvantage by exploring fewer possible solutions. However, considering the cost to rerun the experiment, we still present the results. A future experiment should retest this experiment with the number of unique function evaluations fixed. The number of unique function evaluations is made more consistent for the ring-based TSP experiment.

We chose to use Matlab's SA algorithm to perform the meta-optimization, primarily due to its global search property. Any global search meta-optimization routine could be used instead. Instead of using (8) verbatim, we replace the sums with averages. This allows us to vary $\left|\hat{\mathcal{F}}_+\right|$ and $\left|\hat{\mathcal{F}}_-\right|$ while continuing to give the same relative importance β.

5.2.2 Practices Followed

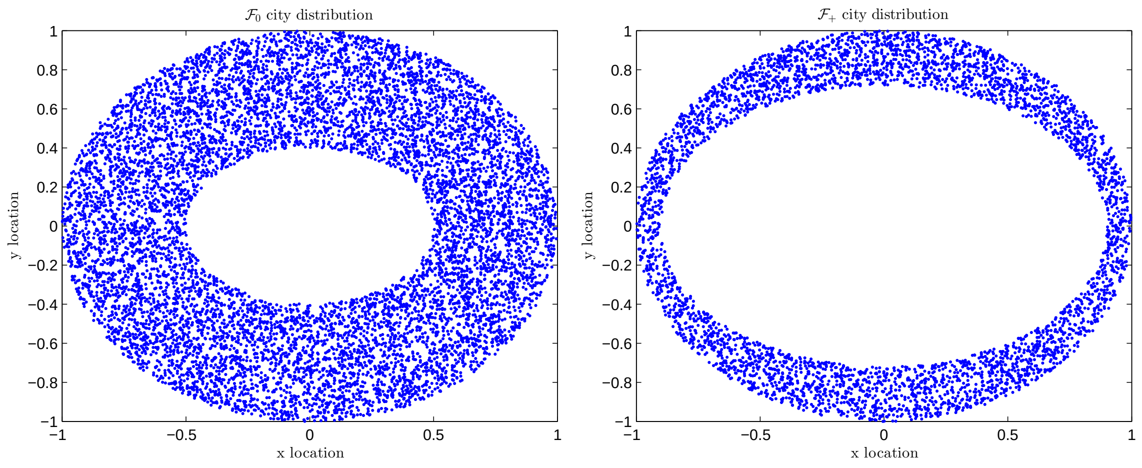
We followed the same practices as discussed in Section 5.1.2.

5.2.3 Trials

We conduct five types of trials, outlined in Table 5: ML5, ML10, ML15, ATSD250+ML5, and ATSD250+ML5 Weak. ML5 can be viewed as the control. We chose the trials ML5, ML10, and ML15 to demonstrate the effects of increasing $\left|\hat{\mathcal{F}}_+\right|$ on a domain where little to no patterns should be exploitable. The ATSD+ML trials, ATSD250+ML5 and ATSD250+ML5 Weak, demonstrate the effects of using random TSP problems for sacrificial data at differing weights of ML compared to ATSD.

Four subtrials are conducted per trial, all using the same $\hat{\mathcal{F}}_+$, $\hat{\mathcal{F}}_-$, or both. The subtrials only differ in the random seed used in the meta-optimization process (*i.e.,* the simulated annealing random seed). This is designed to roughly estimate the variability of the quality of the algorithms returned from the meta-optimization process. Since we hypothesize ATSD and ML should fail, we want to give every chance for ML and ATSD to work. So we use 10,000 simulated annealing iterations to have a greater chance to exploit structure that only works on $\hat{\mathcal{F}}_+$.

We test each resulting algorithm (a total of 20 algorithms) against 10,000 samples of never seen before data from the $\mathcal{F}_+$, $\mathcal{F}_0$, and $\mathcal{F}_-$ distributions.



(a) The $\mathcal{F}_0$ city distribution

(b) The $\mathcal{F}_+$ city distribution

Figure 1: City distributions by problem distribution

5.2.4 Computer setup

The computer setup is very similar to that described in Section 5.1.4. We only use the first two computers as this experiment ran faster, taking only about three days to complete. We use the same Internet based virtual drive for all I/O. The meta-optimization process generates about 963 MB of data including checkpointing every 50 iterations. The code is written to exploit coarse-grain parallelism, scaling up to about 20 instances of Matlab (avoids using the Matlab parallel computing toolbox).

5.3 Traveling Salesperson on a Ring

In this experiment, we investigate ATSD and ML over a discrete optimization problem where multiple properties from the domain should be exploitable. This experiment is in contrast to the high dimensional TSP. As in the previous experiment, we limit the input to only valid sequences of cities. Thus, all input is valid. Only the total distance traveled is analyzed. No information from partial solutions is used. However, rather than generate a correlated cost matrix from random values directly, the cities are randomly placed on a ring and their cost values are computed as the Euclidean distance between cities (*cf.* Figure 1). As the ring's inner and outer diameter become the same, the optimal path converges to traveling in a circular loop. Furthermore, cities are labeled in a clockwise fashion.

The problem distribution comes from the following. $\mathcal{F}_+$ comes from a narrow ring (*cf.* Figure 1b). $\mathcal{F}_0$ comes from a wide ring (*cf.* Figure 1a). $\mathcal{F}_-$ are TSP problems with randomly generated cost matrices, similar to the $\mathcal{F}_-$ in the previous experiment (the high dimensional TSP experiment).

5.3.1 Meta-Optimization Procedure

The meta-optimization steps are similar to the high dimensional TSP's meta-optimization case. We investigate the same hyper-parameters as before: initial paths, mutation rate, max

Table 6: Four Types of Trials

Methods	ML5	ML10	ATSD15+ML5	ATSD50+ML5
$\lvert\hat{\mathcal{F}}_+\rvert$	5	10	5	5
$\lvert\hat{\mathcal{F}}_-\rvert$	0	0	15	50
Subtrials	200	200	200	200
β	0	0	0.2	0.2

swap-size schedule, and swap-size parameter. We use the same custom genetic algorithm as before, with two exceptions. First, we modify the low-level algorithm to force all individuals in the population to represent unique paths. Duplicate paths are replaced with random paths. Because this slows down the low-level algorithm to about half its original speed, we further optimize the low-level algorithm to nearly restore its original performance. We still use Matlab's SA algorithm to find the hyper-parameters. The sums are still replaced with the averages in (8).

5.3.2 Practices Followed

We follow the same practices as discussed in Section 5.1.2.

5.3.3 Trials

We conduct four types of trials, summarized in Table 6: ML5, ML10, ATSD15+ML5, and ATSD50+ML5. ML5 can be viewed as the control. We chose the trials ML5 and ML10 to demonstrate the effects of increasing $\left|\hat{\mathcal{F}}_+\right|$. The ATSD+ML trials, ATSD15+ML5 and ATSD50+ML5, demonstrate the effects of increasing the number of random TSP problems.

200 subtrials are conducted per trial, all using the same $\hat{\mathcal{F}}_+$, $\hat{\mathcal{F}}_-$, or both. The subtrials only differ in the random seed used in the meta-optimization process (*i.e.,* the simulated annealing random seed). This is designed to accurately estimate the variability of the quality of the algorithms returned from the meta-optimization process. Because we are using a total of 800 subtrials, we limit the meta-optimization search to 2,000 simulated annealing iterations for time considerations.

We test each resulting algorithm (a total of 800 algorithms) against 400 samples of never seen before data from the $\mathcal{F}_+$, $\mathcal{F}_0$, and $\mathcal{F}_-$ distributions.

5.3.4 Computer setup

The computer setup is very similar to that described in Section 5.1.4. We only use the first two computers as this experiment took only about one week to complete. We use the same Internet based virtual drive for all I/O. The meta-optimization process generates about 1020 MB of data including checkpointing every 100 iterations. The code is written to exploit coarse-grain parallelism, scaling up to about 800 instances of Matlab. Our parallelism techniques avoid the need for Matlab's parallel computing toolbox.

5.4 Classification Problem

This experiment tests ATSD+ML against a supervised machine learning problem. Specifically, the task is to learn to classify whether an individual makes more than \$50K per year based on 14 attributes such as the sector in which the person works (private, local government, state government, etc.), job type, level of education, sex, age, hours per week, native-country, etc. This data, referred to as the "adult" data, is made publicly available and can be found in the UCI Machine Learning Repository (Bache and Lichman, 2013).

This experiment's goal is to demonstrate how ATSD can be used to boost machine learning performance, not to beat previous classification performance. We compare the classification accuracy of support vector machines (SVMs). The kernel and its parameters are meta-optimized using differing numbers of cross-validation samples and sacrificial problems. Cross-validations are used for $\hat{\mathcal{F}}_+$, which is arguably insufficient (Wolpert, 2001). This is not so much a drawback for the experiment as it demonstrates how quality $\hat{\mathcal{F}}_+$ may be difficult to obtain, further motivating the use of ATSD.

We use three types of sacrificial data. One third of $\hat{\mathcal{F}}_-$ uses real tuples of the attributes, but with random classification results. Another third of $\hat{\mathcal{F}}_-$ uses both random tuples of attributes and classification results. The last third of $\hat{\mathcal{F}}_-$ uses real tuples of attributes, but with a very simple classification problem where the result is wholly determined by a logical conjunction based on the sex, age, and number of years of education. This third set of sacrificial data exhibits several properties absent in the actual data: it is noise free, it depends only on three parameters (ignoring important information such as the job type) and has very low Kolmogorov complexity.

5.4.1 Meta-Optimization Procedure

The meta-optimization objective is to find a good set of SVM hyper-parameters. Specifically, the SVM's kernel, box constraint (training error versus complexity), and kernel parameters (*e.g.* Gaussian kernel width). We investigate seven SVM kernels including:

- Linear: $u^\intercal v$
- Gaussian: $\exp(-|u-v|^2/(2\sigma^2))$
- Laplace: $\exp(-|u-v|/\sigma)$
- Cauchy: $(1+|u-v|^2/\sigma^2)^{-1}$
- Tanh (Multilayer Perception): $\tanh(\alpha u^\intercal v + c)$
- Inhomogeneous polynomial: $(\alpha u^\intercal v + c)^p$
- Logarithmic kernel: $-\log(c+|x-y|^p)$.[9]

σ, α, c, and p in this context are kernel parameters. u and v are data. Note that some of these kernels are only conditionally positive definite. Meaning, if used, the SVM may fail to converge to globally optimal results.

9. Less common kernels were adopted from Cesar Souza's blog: `http://crsouza.blogspot.com/2010/03/kernel-functions-for-machine-learning.html`

The meta-optimization procedure is conducted differently in this experiment than in the other experiments. Unlike the other experiments' meta-optimization procedure, we first sample the algorithm space blindly, then optimize after sampling. This ensures each trial gets the same quality search since they all evaluate the same hyper-parameters. For each kernel, we sample 500 hyper-parameter configurations using low discrepancy Halton sequences (Kuipers and Niederreiter, 1974; Kalos and Whitlock, 2008). This samples the hyper parameters in a roughly uniform manner. All $\hat{\mathcal{F}}_+$ and $\hat{\mathcal{F}}_-$ are evaluated and their results are stored. To evaluate each ML or ATSD trial, we calculate the meta-objective function from the stored results. The hyper-parameters that optimize the meta-objective function for each trial are selected for testing. This makes the search for the optimal algorithm less dependent on a random search (the other experiments used SA). All trials get the same quality search. This also removes the need to test hundreds of different seeds in the meta-optimization process.

When meta-optimizing to select the kernel and kernel parameters that work best for a particular trial, we use a slightly modified (8). Instead of using (8) verbatim, we replace the sums with averages. This allows us to vary $\left|\hat{\mathcal{F}}_+\right|$ and $\left|\hat{\mathcal{F}}_-\right|$ while continuing to give the same relative importance β.

5.4.2 Practices Followed

We mostly follow the same practices as discussed in Section 5.1.2, with one exception. Because we are working with real data in this experiment, we only have a single $f \in \mathcal{F}_+$. So in order to make $\left|\hat{\mathcal{F}}_+\right|$ larger than one, we partition the data into training, validation, and test partitions. We partitioned the data as follows. For the test partition, we used the official test data, `adult_test.csv`, from the UCI Machine Learning Repository (Bache and Lichman, 2013). After removing incomplete data and NaNs, the `adult_test.csv` contains 30162 records. We partition the `adult_data.csv` data into 1/4 for training and 3/4 for validation for each $f \in \mathcal{F}_+$. By making the training data smaller than the validation data, we decreas the time the meta-optimization process uses.

We take the following additional steps to decrease SVM training time. We allow the SVM decision boundary to violate up to 15% of the Karush-Kuhn-Tucker (KKT) conditions (Bazaraa et al., 2006). This means that the result of training the SVM may not be a globally optimal solution, but should in no way invalidate the effects of ATSD. Moreover, we increase the KKT tolerance from 0.001 to 0.05. This means that the KKT conditions close to being satisfied will count as satisfied. This should not invalidate the effects of ATSD, either.

It is trivial to show that relaxing the KKT constraints translates to testing different learning algorithms, other than SVMs. In the NFL framework for supervised learning, each learner is trying to guess the function generating the data (Wolpert, 2001). SVMs trained with relaxed constraints may produce different predictions (*i.e.,* guesses at the function generating the data). This translates as using different learning algorithms than the canonical SVM algorithm. Using a different learning algorithm other than SVMs would have the same consequence. Hence, relaxing the constraints does not invalidate the experiment.

5.4.3 Trials

Because we are meta-optimizing differently, reusing the same function evaluations for each trial, we can test more trials quickly. We vary $\left|\hat{\mathcal{F}}_+\right|$ and $\left|\hat{\mathcal{F}}_-\right|$ with $\beta = 0.1774$ to get 20 trials. We also vary β over zero and 13 logarithmically spaced levels with $\left|\hat{\mathcal{F}}_+\right| = 2$ and $\left|\hat{\mathcal{F}}_-\right| = 4$. Similarly, we vary β over zero and 13 logarithmically spaced levels with $\left|\hat{\mathcal{F}}_+\right| = 4$ and $\left|\hat{\mathcal{F}}_-\right| = 4$. Thus, in total there are 46 unique trials, each denoted using the notation ATSD+X-Y@B. Here $X = \left|\hat{\mathcal{F}}_+\right|$, $Y = \left|\hat{\mathcal{F}}_-\right|$, and $B = \beta$. Trial ATSD+2-12@0.1774 corresponds to the trial with two cross-validations, 12 sacrificial problems, and using $\beta = 0.1774$. When $\beta = 0$, the sacrificial data is given no weight and ATSD+ML degenerates to ML.

5.4.4 Computer setup

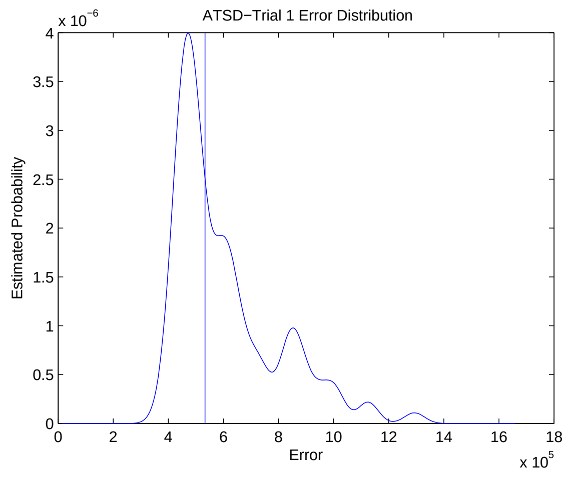
The computer setup is very similar to that described in Section 5.1.4. We only use the first two computers as this experiment takes only about two weeks to complete. We use the same Internet based virtual drive for all I/O. The meta-optimization process generates about 11.9 GB of data, saving all function evaluations (required for optimization after sampling). The code is written to exploit coarse-grain parallelism, scaling up to about 3500 instances of Matlab. Our parallelism techniques avoid the need for Matlab's parallel computing toolbox.

6. Results

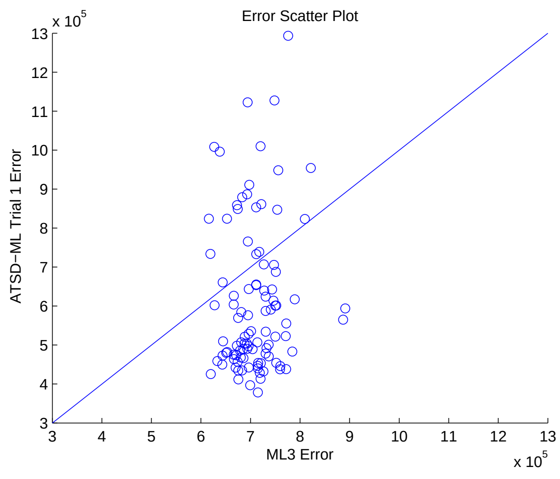
We present the results of the four experiments here. In summary: the satellite trajectory experiment showed ATSD helped, but to lesser extent than ML, as predicted in (12). ATSD also had two of four subtrials drop degrees of freedom in their search. The high dimensional traveling salesperson experiment showed that neither ATSD nor ML have any benefit when no problem knowledge is exploitable. The traveling salesperson on a ring experiment is inconclusive due to an NFL artifact. While the classification problem lacks statistical backing—due to there being a single test problem—some general trends suggest, but are inconclusive, that in the case of a limited search for a better learner, it may be best to limit the sum $\left|\hat{\mathcal{F}}_+\right| + \left|\hat{\mathcal{F}}_-\right|$. Increasing either $\left|\hat{\mathcal{F}}_+\right|$ or $\left|\hat{\mathcal{F}}_-\right|$ after a certain degree produced worse results. A single cross-validation combined with nine sacrificial problems ($\left|\hat{\mathcal{F}}_+\right| = 1, \left|\hat{\mathcal{F}}_-\right| = 9$) outperformed any ML alone.

6.1 Satellite Trajectory Results

We analyze the median performance of our algorithms for two primary reasons. First, we optimized over the median performance. Second, after we collected the data from our experiments, it became evident that the performance distributions of the algorithms are highly skewed (see Figure 2a). Thereupon, we follow the advice of Luke (2013) and Wineberg (2004) and analyze our results using the median rather than the mean.

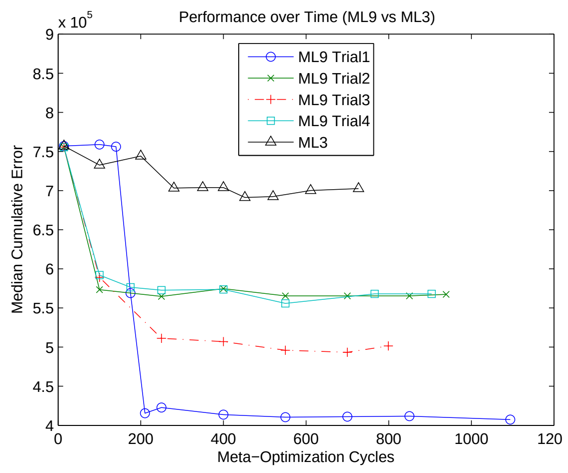


(a) The estimated error distribution

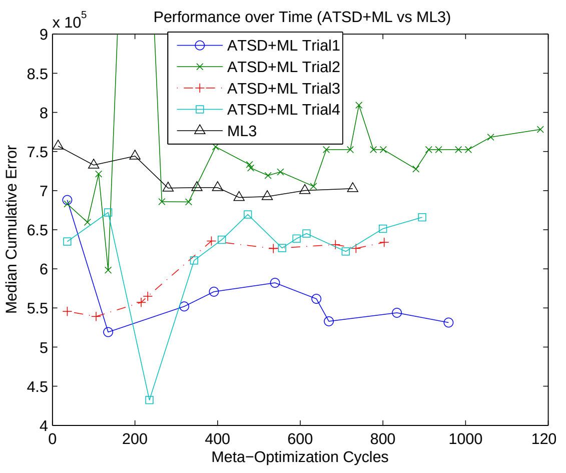


(b) Error from ML3 and ATSD+ML3

Figure 2: Depictions of the cumulative error distribution



(a) ML9 and ML3 performance over time



(b) ATSD+ML3 and ML3 performance over time

Figure 3: Depictions of the cumulative error over time

While we do not analyze the performance over time, we still present the data in Figure 3a and Figure 3b. Notice that while all the trials in Figure 3a start at the same value, the same cannot be said about the trials in Figure 3b (this is due to an unexpected system crash). Thus, the optimization number of function evaluations are incorrect on the ATSD+ML3 trials. This did not influence the final analysis, just the temporal performance seen in Figure 3b. Meta-optimization stopped when Matlab determined insufficient progress was being made.

Table 7 summarizes the median cumulative error of each technique, meaning lower is better. Since there were four trials for each ATSD+ML3 and ML9, to make their analysis easier, we introduced the terms "ATSD+ML3-1," "ATSD+ML3-2," etc. to indicate

Table 7: Summary of Median Performance between Methods

	Median Cumulative Error
Initial	$7.461 \cdot 10^5$
ML3	$7.026 \cdot 10^5$
ATSD+ML3	$6.637 \cdot 10^5$
ML9	$5.249 \cdot 10^5$

particular trials. “ATSD+ML3” and “ML9” are 100 median samples, where the median was taken over the corresponding four trials. Thus, the table reflects the median of medians for ATSD+ML3 and ML9. This table shows that the control performed worst, ML3 was third, ATSD+ML3 was second, and ML9 performed best. We discuss these results after making sure they are statistically significant.

We must analyze the data further to determine if the differences in medians are statistically significant. Due to the great asymmetry, possible dependencies in the data, and multimodal nature of the ATSD+ML3 performance distributions, we avoid using the standard T-tests to compare performance between algorithms. While the Wilcoxon signed rank test is more robust, it is usually employed in the context of symmetric distributions, especially for paired (one-sample) tests (Lehmann, 2004). For this reason, we use the paired sign-test. It usually lacks the power of the other tests but makes fewer assumptions about the data (Lehmann, 2004). Due to the lack of power of the paired sign test, we may permit a lower significance (Lehmann and Romano, 2008).

Table 8 shows the results of the paired sign-tests comparing ATSD+ML3 with ML9. We present the results from the algorithms at the final iteration to reflect the possible effects of overtraining. This table confirms that ML9 outperforms ATSD+ML3: this is to be expected according to theory derived in Section 3.2. According to (12), even with equal compressibility of the meta data and sacrificial data ($\rho_- = \rho_+$), the samples for $\hat{\mathcal{F}}_-$ are expected to be half as efficient.[10] Since ATSD+ML3 uses three samples for $\hat{\mathcal{F}}_+$ and nine samples for $\hat{\mathcal{F}}_-$, according to theory this should perform on par with ML using 7.5 ($3 + 9/2$) samples for $\hat{\mathcal{F}}_+$.

Table 9 shows the results of the paired sign-tests comparing ATSD+ML3 with ML3. This shows that ATSD appears to improve performance, at least on the median. The paired-sign test shows $p = 4.431 \cdot 10^{-2}$, indicating that the null hypothesis (ATSD+ML3 = ML3) is unlikely. Again, the paired-sign test lacks statistical power in general, so $p = 4.431 \cdot 10^{-2}$ is statistically significant.

We also calculated a Bayes factor to directly compare the two hypotheses: median(ML3 − ATSD+ML3) > 0 and median(ML3 − ATSD+ML3) < 0. The difference between ML3 and ATSD+ML3 is distributed approximately as an extreme-value distribution (*cf.* Figure 4).[11] Matlab's extreme-value distribution has its median at $\mu + \sigma \log(\log(2))$. Thus, for our first hypothesis, we integrated over the range of parameters where $\mu > -\log(\log(2))\sigma$ and for the second hypothesis we used $\mu < -\log(\log(2))\sigma$. Since Bayes factors use a prior

10. Our $\hat{\mathcal{F}}_-$ comes from a more complicated distribution, so we expect a lower ρ_- than ρ_+. This is supported by the fact that ATSD+ML3 performs more similarly to ML3 than ML9.
11. We have no reason to believe the data should be distributed as such, but extreme-value distributions provided the best fit.

Table 8: Sign Test (H_a: ATSD+ML3 > ML9)

	ML9-1	ML9-2	ML9-3	ML9-4	ML9
ATSD+ML3-1	$1.30 \cdot 10^{-12}$	0.972	$6.02 \cdot 10^{-3}$	0.903	0.382
ATSD+ML3-2	$7.97 \cdot 10^{-29}$	$6.55 \cdot 10^{-12}$	$1.60 \cdot 10^{-19}$	$1.35 \cdot 10^{-10}$	$1.53 \cdot 10^{-17}$
ATSD+ML3-3	$1.32 \cdot 10^{-25}$	$9.05 \cdot 10^{-8}$	$6.26 \cdot 10^{-23}$	$9.05 \cdot 10^{-8}$	$1.00 \cdot 10^{-21}$
ATSD+ML3-4	$1.27 \cdot 10^{-16}$	$2.04 \cdot 10^{-4}$	$5.58 \cdot 10^{-10}$	$2.04 \cdot 10^{-4}$	$2.76 \cdot 10^{-8}$
ATSD+ML3	$1.32 \cdot 10^{-25}$	$1.35 \cdot 10^{-10}$	$1.32 \cdot 10^{-25}$	$2.41 \cdot 10^{-13}$	$6.26 \cdot 10^{-23}$

Table 9: Sign Test (H_a: ATSD+ML3 < ML3)

	ML3
ATSD+ML3-1	$9.050 \cdot 10^{-8}$
ATSD+ML3-2	0.9334
ATSD+ML3-3	$1.759 \cdot 10^{-3}$
ATSD+ML3-4	0.2421
ATSD+ML3	$4.431 \cdot 10^{-2}$

distribution, we chose μ to be distributed normally about σ with a standard deviation corresponding to the range of Matlab's `evfit`'s 95% confidence interval. σ is distributed half-normal with a standard deviation corresponding to the range of its 95% confidence interval. The resulting Bayes factor, $k = 5.571$, means that it is more than five times as likely that ATSD+ML3 has a lower (better) median than ML3. According to Robert et al. (2009), this is *substantial* evidence.

There were two surprising, unanticipated results that only occurred for the experiments using anti-training with sacrificial data combined with meta-learning (ATSD+ML3). Two out of the four ATSD+ML3 meta-optimization runs (trials 2 and 3) halted early with the message "distance between lower and upper bounds, in dimension 4 is too small to compute finite-difference approximation of derivative." This corresponds to the two runs limiting their search by dropping a degree of freedom corresponding to the elliptical orientation of the satellite's orbit.[12] This is plausible: the debris in $\mathcal{F}_+$ have no large angular gaps, but the debris in $\mathcal{F}_-$ did have angular gaps. Thus, on average no orientation is to be preferred on $\mathcal{F}_+$, but some orientations are significantly worse for $\mathcal{F}_-$. Recall that in general ATSD+ML maximizes the performance difference between $\mathcal{F}_+$ and $\mathcal{F}_-$. As such, we suspect that ATSD+ML3 extracted this information from this difference. It seems unlikely that this result occurred due to random chance as two trials from ATSD+ML3 dropped this degree of freedom, but none of the five ML did so.

Another surprise is that the second ATSD+ML3 subtrial also dropped a second degree of freedom from its search: the initial true anomaly. Dropping this variable appears to be a valid choice for speeding the search for good solutions, albeit less obvious. Provided that the ratio of orbital periods are irrational (or requiring very large rational numbers) and ignoring highly eccentric orbits, the equidistribution theorem can be applied to show that

12. Meta-optimization was allowed to continue with the variable replaced by the average of the lower and upper bounds.

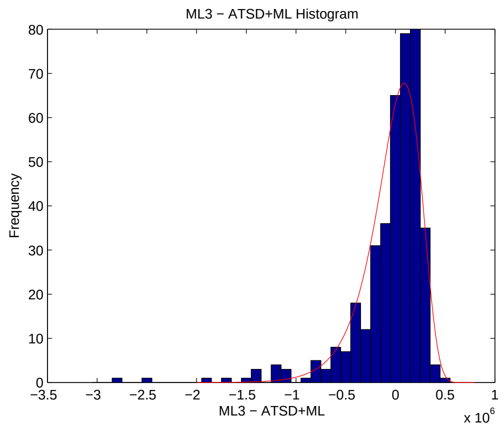


Figure 4: ML3 $-$ ATSD+ML3 and its best fit distribution

the difference in orbital-phases will be uniform (or nearly uniform). Thus, provided enough time, the initial true anomaly may also be ignored. How can we reconcile this with the fact that the second ATSD+ML3 trial performed the worst out of all the ATSD+ML trials? We do not know the answer, but speculate it is because we only tested the performance over $\hat{\mathcal{F}}_+$. It is possible that this reduced degree of freedom would be beneficial in the more general $\mathcal{F}_+ \cup \mathcal{F}_0$. Further testing is required.

6.2 Random TSP Results

As expected, neither ML nor ATSD produce optimization algorithms that reduce error when no problem structure may be exploited. Table 10 shows the mean and standard deviation (taken across the four subtrials) of the mean cumulative error across 10000 tests. Notice that the cumulative error for ML15 over $\mathcal{F}_+$ is worse than ML10's error, although by a statistically insignificant amount. ML10's improvement over ML5's error is also statistically insignificant ($p = 0.4315$). The evidence suggests that increasing the number of meta-learning samples has an insignificant effect on algorithm performance. Similarly, the two ATSD tests (ATSD250+ML5 and ATSD250+ML5 Weak) show no improvement over ML5 alone. They actually show higher mean error over $\mathcal{F}_+$. This could be due to random chance or by the fact that this version of the GA optimizer permits duplicate function evaluations, putting those trials at a disadvantage.[13]

Another issue of concern is whether the meta-optimization procedure succeeded in optimizing (8). Note that we replaced the sums with averages, so β reflects the relative weight of ATSD compared to ML. This is used in Table 11 for the meta-fitness column. It shows how well the discovered algorithms perform according to their meta-optimization criteria. The ATSD250+ML5 trial has the most negative meta-fitness when $\beta = 1.0$, meaning it provides the better solution to (8) than any other algorithms investigated. However, ATSD250+ML5 Weak which was optimized with respect to $\beta = 0.2$, fails to have the lowest meta-fitness for that column. This means that the algorithms discovered

13. The following experiment, the TSP problem on a ring, forces the GA optimizer to use unique function evaluations within each iteration.

Table 10: Mean and Standard Deviation of Means

Trial	$\mathcal{F}_-$		$\mathcal{F}_0$		$\mathcal{F}_+$	
	μ	σ	μ	σ	μ	σ
ML5	47.20	0.4175	47.37	0.3774	47.12	0.3710
ML10	47.12	0.3888	47.28	0.3629	47.07	0.3642
ML15	47.13	0.5617	47.31	0.5370	47.07	0.5196
ATSD250+ML5	49.83	1.981	49.90	1.870	49.56	1.811
ATSD250+ML5 Weak	48.35	1.043	48.47	0.9683	48.18	0.9645

Table 11: Meta Objective Function Values

Trial	mean meta-fitness ($\beta = 1.0$)	mean meta-fitness ($\beta = 0.2$)
ML5	$-7.972 \cdot 10^{-2}$	37.68
ML10	$-4.700 \cdot 10^{-2}$	37.65
ML15	$-5.931 \cdot 10^{-2}$	37.65
ATSD250+ML5	-0.2708	39.59
ATSD250+ML5 Weak	-0.1695	38.51

in ATSD250+ML5 Weak are, even according to theory, inferior to the algorithms from the ML15 trials. This is a side-effect of the NFL theorems: it may be better to search for something else rather than the desired objective function.

6.3 TSP Ring Results

Because this experiment uses a large number of subtrials, we can illustrate the differential error distributions. Figure 5 shows how the ATSD15+ML5 and ATSD50+ML5 trials compare to the ML5 and ML10 trials. Since these distributions are symmetric with few outliers, it makes sense to use the standard T-test to compare means. Furthermore, since the individual test problems are the same for each algorithm, we are permitted to use the paired T-test. Negative values favor ATSD and positive values favor ML.

Contrary to what we anticipated, all the distributions in Figure 5 have a positive average, meaning the algorithms found from the ML trials are to be preferred. The mean error and its standard deviation for each trial over each problem distribution $\mathcal{F}_-, \mathcal{F}_0, \mathcal{F}_+$ are shown in Table 12. Table 13 shows the results of the T-tests, where the alternative hypotheses are that ATSD15+ML5 and ATSD50+ML5 have greater means than ML5 and ML10.[14] Since all the p values are sufficiently small, the null hypotheses should be rejected in favor of the alternatives. The algorithms found in the ATSD15+ML5 and ATSD50+ML5 trials perform worse than the algorithms found in the ML5 and ML10 trials.

Even though the algorithms found in the ATSD trials perform statistically significantly worse than the algorithms found in the ML trials, these results do not contradict our theory. To see why these results do not contradict our theory, we need to investigate the meta-objective function values. These values are shown in Table 14. The meta-optimization

14. The significances in Table 13 were so small, arbitrary precision arithmetic was required to compute them.

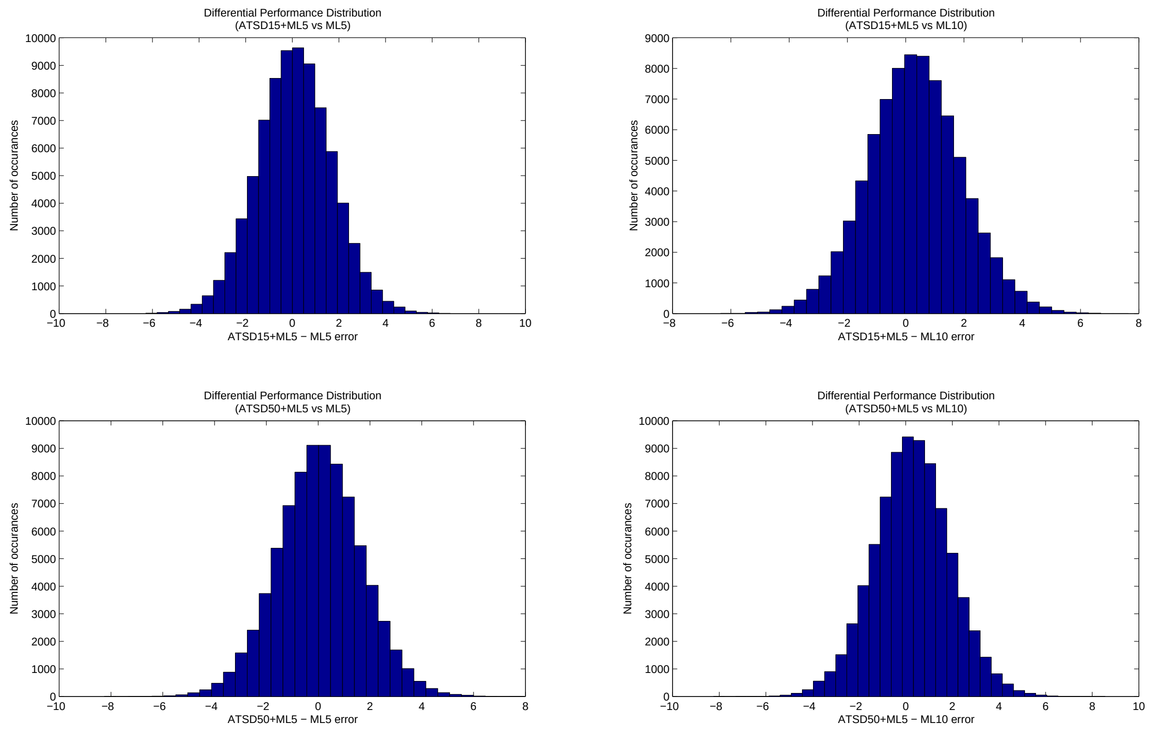


Figure 5: Error differences as a distribution

Table 12: Mean and Standard Deviation of Means

Trial	$\mathcal{F}_-$		$\mathcal{F}_0$		$\mathcal{F}_+$	
	μ	σ	μ	σ	μ	σ
ML5	17.29	1.210	15.57	1.309	16.96	1.438
ML10	17.19	1.216	15.39	1.309	16.74	1.448
ATSD15+ML5	17.37	1.212	15.65	1.304	17.05	1.427
ATSD50+ML5	17.35	1.221	15.62	1.323	17.02	1.453

Table 13: Significance of Differential Performance

Distribution	Significance
ATSD15+ML5 - ML5	$3.783 \cdot 10^{-63}$
ATSD15+ML5 - ML10	$1.150 \cdot 10^{-642}$
ATSD50+ML5 - ML5	$9.566 \cdot 10^{-30}$
ATSD50+ML5 - ML10	$7.845 \cdot 10^{-508}$

Table 14: Meta Objective Function Values when $\beta = 0.2$

Trial	mean meta-fitness
ML5	13.50
ML10	13.30
ATSD15+ML5	13.58
ATSD50+ML5	13.55

for ML5 and ML10 found better solutions for the ATSD+ML meta-objective function than the ATSD15+ML5 and ATSD50+ML5 trials! This is a side-effect of the NFL theorems. Without knowing anything about the search for optimization algorithms, we naïvely used a SA search. Using SA to reduce the meta-objective function with $\beta = 0.2$ actually performed worse at reducing it than when trying to reduce meta-objective function with $\beta = 0.0$ (ML alone). To better understand what happened consider the following analogy. When a dog (an algorithm) was told to find (optimize) the yellow ball (the ATSD objective function), it found nothing. However, when told to find the violet ball (the ML objective function), it found both the yellow and violet balls (good solutions to both meta-optimization functions).

So while this experiment does not confirm our theory, neither does the experiment deny it. Instead, it provided meaningful feedback. Matlab's SA algorithm performs better on ML than ATSD+ML for the small set of problems investigated in this experiment. We can prevent this problem from occurring by ensuring all trials evaluate an identical set of algorithms. This effectively amounts to sampling algorithms first, recording their performance, then optimizing after all algorithms have been tested on all problems. Thus, it is impossible for ML to find an algorithm that outperforms ATSD+ML on ATSD+ML's own objective function. If ML were to find such an algorithm, because ATSD+ML would test it too, ATSD+ML must at least match it (assuming the function values are deterministic). The following experiment uses this adjustment.

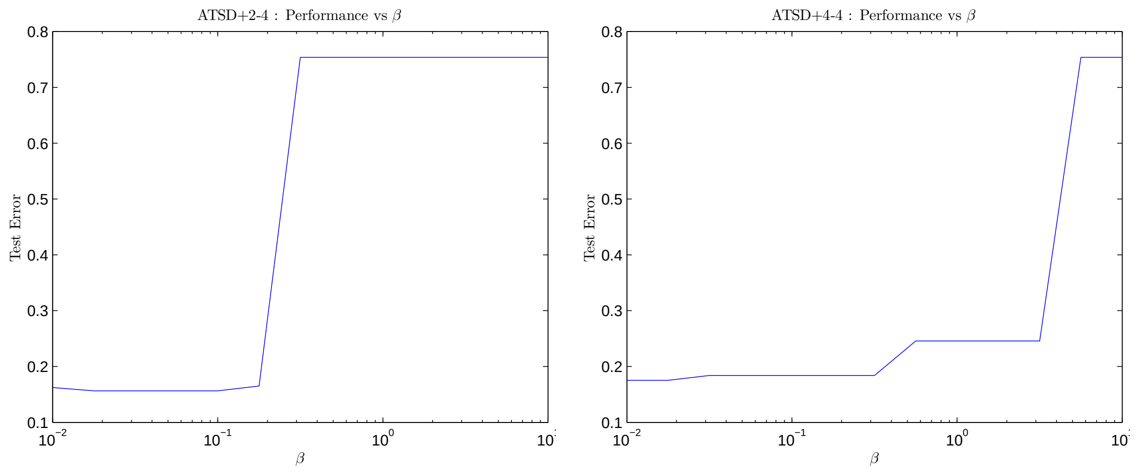
6.4 Classification Results

The results from this experiment lack statistical backing since we use only a single test problem. We considered using bootstrapping to estimate the variance of the error rates presented in this experiment but ultimately avoided it due to time considerations—the experiment already took two weeks to complete. A ten-fold cross-validation would take much longer than an additional two weeks.[15] Future work will include bootstrapping to estimate the variance in this experiment.

Despite this, some general trends are still evident. We modified (8) by replacing the sums with averages so β reflects the relative weight of ATSD. Figures 6a and 6b show that large values of β cause more harm than good. Figure 6a shows $2 \cdot 10^{-2} < \beta < 10^{-1}$ as being ideal, whereas Figure 6b shows that ATSD should be avoided. Since this difference in behavior could be noise, we further investigated the error rate's behavior.

Table 15 shows several patterns. First, in terms of error, ATSD+1-9@0.1778 and ATSD+1-12@0.1778 perform best. These two trials even outperform the best performing ML (ATSD+3-0@0). Another point is that the best performing trials have a negatively

15. Although, it should take less than twenty weeks to complete a ten-fold cross validation study.



(a) Error rate when $\left|\hat{\mathcal{F}}_+\right| = 2$ and $\left|\hat{\mathcal{F}}_-\right| = 12$ (b) Error rate when $\left|\hat{\mathcal{F}}_+\right| = 4$ and $\left|\hat{\mathcal{F}}_-\right| = 12$

Figure 6: Error rate versus β

Table 15: Error versus $\left|\hat{\mathcal{F}}_+\right|$ and $\left|\hat{\mathcal{F}}_-\right|$ with β=0.1778

	$\left\|\hat{\mathcal{F}}_-\right\|$=0	$\left\|\hat{\mathcal{F}}_-\right\| = 3$	$\left\|\hat{\mathcal{F}}_-\right\| = 6$	$\left\|\hat{\mathcal{F}}_-\right\| = 9$	$\left\|\hat{\mathcal{F}}_-\right\| = 12$
$\left\|\hat{\mathcal{F}}_+\right\|$=1	0.1624	0.1624	0.1624	**0.1558**	**0.1558**
$\left\|\hat{\mathcal{F}}_+\right\|$=2	0.1624	0.1629	0.1624	**0.1564**	0.1652
$\left\|\hat{\mathcal{F}}_+\right\|$=3	**0.1564**	0.1579	0.1579	**0.1564**	0.1839
$\left\|\hat{\mathcal{F}}_+\right\|$=4	**0.1752**	**0.1752**	0.1839	0.1839	0.1839

correlated $\left|\hat{\mathcal{F}}_-\right|$ and $\left|\hat{\mathcal{F}}_+\right|$. The bold diagonal highlights this. This pattern lacks statistical significance, thus one should not read into it too much. Figure 7 also visualizes these results.

While we are not trying to beat historical classification performance on this data set, it is worthwhile to note preexisting performance on this data set. Kohavi (1996) used naïve Bayes classification trees to produce predictions with errors ranging from about 15.7% to 16.7%, and the C4.5 algorithm produced predictions with errors ranging from about 13.7% to 14.8%.

7. Discussion and Conclusion

We used the statistical entropy of the objective function distributions to address when meta-learning (ML) is possible. For ML to benefit learning, the typical problems should have low Kolmogorov complexity. This implies that it is necessary for the problem distribution to be at least compressible, but not necessarily fit in polynomial space.

We introduced a novel theoretical approach, anti-training with sacrificial data, to improve machine-learning performance. ATSD directly follows from and relies on the No Free Lunch (NFL) theorems. It can be seen as an approximate dual of ML. As

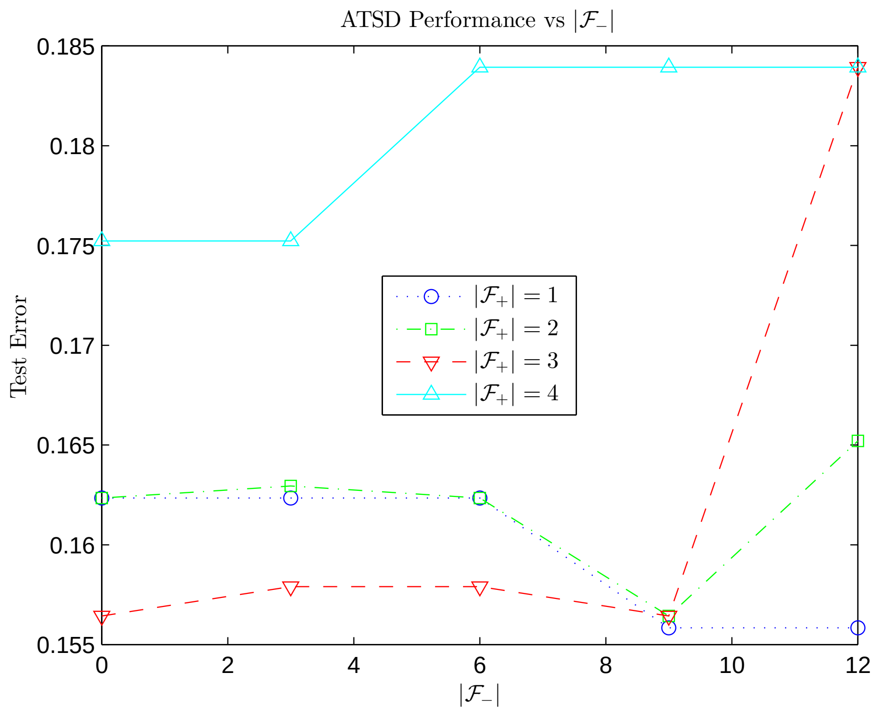


Figure 7: Error rate versus $\left|\hat{\mathcal{F}}_-\right|$

such, the theory dictates that ATSD works where ML works, namely when the probable problems are compressible. While ML directly improves performance over a set of common problems, ATSD worsens performance over sacrificial (impossible and unlikely) problems. Due to the NFL theorems, performance must increase over the remaining sum of all other problem, although it is possible for performance to transfer from the typical problems to any unspecified problems. To help counter-balance this possibility, ATSD ought to be combined with ML (ATSD+ML).

We examined the theoretical efficiency of ATSD using a deck of cards analogy. We arrived at a function relating the percent of implausible functions, the compressibility of typical functions, and the compressibility of implausible functions with the efficiency of ATSD. Assuming that about half of all functions are exceedingly unlikely and the ML objective functions are as compressible as the sacrificial objective functions, then the relative efficiency of ATSD is half as efficient as ML. Moreover, we addressed how to generate sacrificial data and qualitatively assess their compressibility.

ATSD+ML has other helpful properties. Since sacrificial data is easy to generate, it can be applied even when real data is scarce. The theory suggests this is when ATSD+ML is expected to be most effective. The fourth experiment provides supportive, but inconclusive evidence of this. According to the theory, it also helps prevent overtraining. By providing an empirical definition of noise, theoretically the learning algorithm can fail to learn noise. Also when combined with ML, ATSD+ML makes it less likely to overfit the original ML objective function. ATSD+ML also allows for the specification of failure cases, reducing the number of unexpected failures.

ATSD+ML is a technique still in its infancy. Much work remains to be done on accurately estimating how much and what kind of sacrificial data ATSD+ML needs. We only touched on the proper balance between ATSD and ML. In general, a safe value for β in (8) is $0 \leq \beta < \left|\hat{\mathcal{F}}_+\right| / \left|\hat{\mathcal{F}}_-\right|$, or when the sums are replaced with averages $0 \leq \beta < 1$. Another future research direction includes studying the implications of the NFL theorems approximately holding, when the equality in (1) is replaced with upper and lower bounds. We conjecture that the benefits from anti-training would diminish as the equality becomes less strict. However, this would need to be investigated more thoroughly. ATSD+ML could be viewed as a form of regularization and studied as such. It might also be able to be applied to model selection rather than algorithm selection. A probabilistic model would be useful in answering these questions.

The basic concept of ATSD—improving performance by training to perform worse on irrelevant data—may have benefits for selecting solutions (not algorithms) to improve predictive accuracy. This baseline ATSD does not follow from the NFL theorems in the same manner. It may still follow from the NFL theorems, but for a different reason. However, baseline ATSD may be justifiable in terms of regularization. If a solution fits random noise in addition to the actual data, then the solution may be unnecessarily complex.

We conducted four experiments showing how ATSD may function in practice. The first experiment produced results showing ATSD improves performance for optimizers. The second experiment confirmed that ATSD should fail when ML fails. The third experiment was inconclusive as the algorithms discovered from ML outperformed the ATSD candidates even at their own meta-objective function; a side effect of the NFL algorithms. The fourth experiment provides supportive, but inconclusive evidence that ATSD benefits classification algorithms. More experiments are left for future work. Such experiments include repeating similar experiments that take significantly less time to complete, using 10-fold cross-validation for machine learning experiments, and comparing direct meta-optimization (as in the first three experiments) to sampling based approached (as in the fourth experiment).

Convex optimization is a desirable property needed for more sophisticated machine learning procedures. This is the issue when extending ATSD to work with support-vector machines, support-vector regression, or any other optimization that depends on convex optimization problems. The results from the third experiment motivate a convex meta-optimization framework to ensure algorithms actually optimize their meta-objective function. The issue is that it is possible for the global minimum to occur whenever the sacrificial performance approaches positive infinity; this drives the objective function to negative infinity. This in turn causes the traditional objective function to be largely ignored. The most natural approach would be to apply some convex non-decreasing saturating function (*e.g.,* $exp(-x)$) to the sacrificial term. The effects of this saturating ATSD on convex optimization techniques should be studied, as well as other techniques to extend ATSD to a convex optimization framework.

References

Sylvain Arlot and Alain Celisse. A survey of cross-validation procedures for model selection. *Statistics Surveys*, 4:40–79, 2010. doi: 10.1214/09-SS054. URL `http://arxiv.org/abs/0907.4728`.

Peter Auer, Mark Herbster, and Manfred K. Warmuth. Exponentially many local minima for single neurons, 1995.

Anne Auger and Olivier Teytaud. Continuous lunches are free! In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, GECCO '07, pages 916–922, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-697-4. doi: 10.1145/1276958.1277145. URL `http://doi.acm.org/10.1145/1276958.1277145`.

Anne Auger and Olivier Teytaud. Continuous lunches are free plus the design of optimal optimization algorithms. *Algorithmica*, 57:121–146, 2010. ISSN 0178-4617. doi: 10.1007/s00453-008-9244-5. URL `http://dx.doi.org/10.1007/s00453-008-9244-5`.

K. Bache and M. Lichman. UCI machine learning repository, 2013. URL `http://archive.ics.uci.edu/ml`.

Mokhtar S. Bazaraa, Hanif D. Sherali, and C. M. Shetty. *Nonlinear Programming: Theory and Algorithms*, chapter 4, pages 188–195. Wiley, 3rd edition, 2006.

J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *The Journal of Machine Learning Research*, 13:281–305, 2012.

W. Bialek, I. Nemenman, and N. Tishby. Predictability, complexity, and learning. *Neural Computation*, 13(11):2409–2463, 2001.

Peter J Brockwell and Richard A Davis. *Introduction to time series and forecasting*. Springer Verlag, 2002. ISBN: 9780387953519.

D. Corne and J. Knowles. No free lunch and free leftovers theorems for multiobjective optimisation problems. In *Evolutionary Multi-Criterion Optimization*, pages 66–66. Springer, 2003a. URL `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.15.1480`.

D. Corne and J. Knowles. Some multiobjective optimizers are better than others. In *Proceedings Congress Evolutionary Computation CEC '03*, volume 4, pages 2506–2512, 2003b. doi: 10.1109/CEC.2003.1299403.

T.M. Cover and J.A. Thomas. *Elements of Information Theory*, chapter Kolmogorov Complexity, pages 463–508. John Wiley & Sons, Inc., Hoboken, New Jersey, second edition, 2006. ISBN 978-0471241959.

Joseph C. Culberson. On the futility of blind search: An algorithmic view of 'no free lunch'. *Evolutionary Computation*, 6:109–127, June 1998. ISSN 1063-6560. doi: http://dx.doi.org/10.1162/evco.1998.6.2.109. URL `http://dx.doi.org/10.1162/evco.1998.6.2.109`.

Stefan Droste, Thomas Jansen, and Ingo Wegener. Optimization with randomized search heuristics – the (a)nfl theorem, realistic scenarios, and difficult functions. *Theoretical Computer Science*, 287(1):131–144, 2002. URL `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.35.5850`.

N. Gershenfeld. *The nature of mathematical modeling*, chapter Linear and Nonlinear Time Series, pages 204–224. Cambridge University Press, 1999.

C. Giraud-Carrier and F. Provost. Toward a justification of meta-learning: Is the no free lunch theorem a show-stopper? In *Proceedings of the ICML-2005 Workshop on Meta-learning*, pages 12–19, Bonn, Germany, 2005. URL `http://dml.cs.byu.edu/~cgc/pubs/ICML2005WS.pdf`.

David S. Goodsell and Arthur J. Olson. Automated docking of substrates to proteins by simulated annealing. *Proteins: Structure, Function, and Bioinformatics*, 8(3):195–202, 1990. ISSN 1097-0134. doi: 10.1002/prot.340080302. URL `http://dx.doi.org/10.1002/prot.340080302`.

Christian Igel and Marc Toussaint. On classes of functions for which no free lunch results hold. *Information Processing Letters*, 86(6):317–321, June 2003. ISSN 0020-0190. doi: 10.1016/S0020-0190(03)00222-9. URL `http://dx.doi.org/10.1016/S0020-0190(03)00222-9`.

Christian Igel and Marc Toussaint. A no-free-lunch theorem for non-uniform distributions of target functions. *Journal of Mathematical Modelling and Algorithms*, 3:2004, 2004. URL `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.71.8446`.

L. Ingber. Simulated annealing: Practice versus theory. *Mathematical and Computer Modelling*, 18(11):29 – 57, 1993. ISSN 0895-7177. doi: http://dx.doi.org/10.1016/0895-7177(93)90204-C. URL `http://www.sciencedirect.com/science/article/pii/089571779390204C`.

Lester Ingber. Adaptive simulated annealing (asa): Lessons learned. *Control and Cybernetics*, 25:33–54, 1996.

Malvin H Kalos and Paula A Whitlock. *Monte carlo methods*, chapter Quasi-Monte Carlo, pages 101–103. John Wiley & Sons, second edition, 2008.

Ron Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the 14th international joint conference on Artificial intelligence - Volume 2*, IJCAI'95, pages 1137–1143, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc. ISBN 1-55860-363-8. URL `http://dl.acm.org/citation.cfm?id=1643031.1643047`.

Ron Kohavi. Scaling up the accuracy of naive-bayes classifiers: a decision-tree hybrid. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pages 202–207, Menlo Park, California, 1996. AAAI Press.

Lauwerens Kuipers and Harald Niederreiter. *Uniform distribution of sequences*, chapter Special Sequences, pages 129–130. Courier Dover Publications, 1st edition, 1974. URL `http://web.maths.unsw.edu.au/~josefdick/preprints/KuipersNied_book.pdf`.

E.L. Lehmann. *Elements of Large-Sample Theory*, chapter 3.4 Comparison of tests: Relative efficiency, pages 173–187. Springer, 2004.

E.L. Lehmann and Joseph P. Romano. *Testing statistical hypotheses*, chapter 3.1 Stating The Problem, pages 56–59. Springer, Spring Street, New York, NY 10013, USA, 2008.

Ming Li and Paul Vitányi. *An introduction to Kolmogorov complexity and its applications*, chapter Algorithmic Complexity, pages 101 – 107. Springer, Spring Street, New York, NY, 3rd edition, 2008. doi: 10.1007/978-0-387-49820-1.

Sean Luke. *Essentials of Metaheuristics*. Lulu, second edition, 2013. URL `http://cs.gmu.edu/~sean/book/metaheuristics/Essentials.pdf`. Available for free at http://cs.gmu.edu/∼sean/book/metaheuristics/.

M.D. McKay, R.J. Beckman, and W.J. Conover. Comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21(2):239–245, 1979.

John R. Rice. The algorithm selection problem. *Advances in Computers*, 15:65–118, 1976. URL `http://www.cs.purdue.edu/research/technical_reports/1975/TR%2075-152.pdf`.

J. Rissanen. Universal coding, information, prediction, and estimation. *Information Theory, IEEE Transactions on*, 30(4):629 – 636, jul 1984. ISSN 0018-9448. doi: 10.1109/TIT.1984.1056936.

J. Rissanen. Stochastic complexity and modeling. *The Annals of Statistics*, pages 1080–1100, 1986.

J. Rissanen. Stochastic complexity. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 223–239, 1987.

J. Rissanen. *Stochastic complexity and statistical inquiry.* World Scientific, Singapore, 1989.

Christian P. Robert, Nicolas Chopin, and Judith Rousseau. Harold jeffreyss theory of probability revisited. *Statistical Science*, 24(2):141–172, 2009. doi: 10.1214/09-STS284.

C. Schumacher, M.D. Vose, and L.D. Whitley. The no free lunch and problem description length. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 565–570, 2001.

Robert H Shumway and David S Stoffer. *Time series analysis and its applications: with R examples.* Springer, New York, Dordrecht, Heidelberg, London, 3rd edition, 2011.

Ricardo Vilalta and Youssef Drissi. A perspective view and survey of meta-learning. *Artificial Intelligence Review*, 18:77–95, 2002.

Guan Ming Wang, Estela Blaisten-Barojas, A. E. Roitberg, and T. P. Martin. Strontium clusters: Many-body potential, energetics, and structural transitions. *The Journal of Chemical Physics*, 115(8):3640–3646, 2001. doi: 10.1063/1.1384454. URL `http://link.aip.org/link/?JCP/115/3640/1`.

Darrell Whitley and Jean Watson. Complexity theory and the no free lunch theorem. In Edmund K. Burke and Graham Kendall, editors, *Search Methodologies*, pages 317–339. Springer US, 2005. ISBN 978-0-387-28356-2. doi: 10.1007/0-387-28356-0_11. URL `http://dx.doi.org/10.1007/0-387-28356-0_11`.

Mark Wineberg. Introductory statistics for evolutionary computation. In *The Sixth Genetic and Evolutionary Computation Conference (GECCO 2004)*, 2004.

David H. Wolpert. The supervised learning no-free-lunch theorems. In *Proceedings 6th Online World Conference on Soft Computing in Industrial Applications*, pages 25–42, 2001. URL `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.99.133`.

D.H. Wolpert and W.G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67 –82, apr 1997. ISSN 1089-778X. doi: 10.1109/4235.585893.

D.H. Wolpert and W.G. Macready. Coevolutionary free lunches. *IEEE Transactions on Evolutionary Computation*, 9(6):721–735, dec. 2005. ISSN 1089-778X. doi: 10.1109/TEVC.2005.856205.

Xiaoguang Zhang, Li Yu, Yuan Zheng, Yu Shen, Guangtao Zhou, Lin Chen, Lixia Xi, Tiecheng Yuan, Jianzhong Zhang, and Bojun Yang. Two-stage adaptive pmd compensation in a 10 gbit/s optical communication system using particle swarm optimization algorithm. *Optics Communications*, 231(1–6):233 – 242, 2004. ISSN 0030-4018. doi: 10.1016/j.optcom.2003.12.045. URL `http://www.sciencedirect.com/science/article/pii/S003040180302385X`.

Appendix A. Proof of the two-valued nature of $\hat{P}_+(f)$ in practice

The estimate of $P_+(f)$ is denoted $\hat{P}_+(f)$. We argue that $\hat{P}_+(f)$ is practically going to be a two-valued function: 0 and $1/\left|\hat{\mathcal{F}}_+(f)\right|$. This is demonstrated by showing that when independently sampling from $\mathcal{F}_+$, the probability that any two functions are identical is exceedingly small in practice. If one stops to ponder for a moment about similar problems, one realizes this problem reduces to the well known birthday problem in probability theory.

The birthday problem asks "what is the probability that out of n people (samples), some pair of them will have the same birthday (function)." The answer is 100% if the number of people exceed the number of days in a year, but otherwise it is:

$$1 - \frac{n!\binom{365}{n}}{365^n}.$$

When replacing days with functions and people with samples this becomes

$$1 - \frac{n!\binom{\left|\mathcal{F}_+\right|}{n}}{\left|\mathcal{F}_+\right|^n}. \tag{13}$$

While (13) is exact, it must be used iteratively to figure out how much larger $\left|\mathcal{F}_+\right|$ must be than n to ensure a small probability of failure (that two functions will be identical). We provide an approximate n which limits the probability of failure to ϵ. Consider the probability of two samples having different functions,

$$\left(1 - \frac{1}{\left|\mathcal{F}_+\right|}\right).$$

If there are n samples, then there exist $n(n-1)/2$ pairs of samples. Assuming each comparison is independent (this is approximately true if $n \ll \left|\mathcal{F}_+\right|$), then the probability that all samples are unique is

$$\left(1 - \frac{1}{\left|\mathcal{F}_+\right|}\right)^{n(n-1)/2}$$

Using the binomial theorem, this expands to

$$1 - \binom{\frac{n^2-n}{2}}{1}\frac{1}{\left|\mathcal{F}_+\right|} + \binom{\frac{n^2-n}{2}}{2}\frac{1}{\left|\mathcal{F}_+\right|^2} - \binom{\frac{n^2-n}{2}}{3}\frac{1}{\left|\mathcal{F}_+\right|^3} + \binom{\frac{n^2-n}{2}}{4}\frac{1}{\left|\mathcal{F}_+\right|^4} - \dots. \tag{14}$$

We will bound the lth term of (14) using the well known inequality $\binom{n}{l} \leq \frac{n^l}{l!}$ to get

$$\begin{aligned}
\binom{\frac{n^2-n}{2}}{l}\frac{1}{\left|\mathcal{F}_+\right|^l} &\leq \frac{\left(\frac{n^2-n}{2}\right)^l}{l!}\frac{1}{\left|\mathcal{F}_+\right|^l} \\
&\leq \left(\frac{n^2-n}{2}\right)^l \frac{1}{\left|\mathcal{F}_+\right|^l} \\
&= \left(\frac{n^2-n}{2\left|\mathcal{F}_+\right|}\right)^l.
\end{aligned}$$

Thus, all higher order terms are negligible so long as

$$\frac{n^2 - n}{2\left|\mathcal{F}_+\right|} \ll 1$$

The first two terms are kept and set equal to the desired level of probability

$$\begin{aligned} 1 - \binom{\frac{n^2-n}{2}}{1} \frac{1}{\left|\mathcal{F}_+\right|} &\geq 1 - \epsilon \\ \binom{\frac{n^2-n}{2}}{1} \frac{1}{\left|\mathcal{F}_+\right|} &\leq \epsilon \\ \frac{n^2 - n}{2\left|\mathcal{F}_+\right|} &\leq \epsilon \\ n^2 - n &\leq 2\left|\mathcal{F}_+\right| \epsilon \\ n \leq \frac{\sqrt{8\left|\mathcal{F}_+\right| \epsilon + 1} + 1}{2} & \qquad (15) \end{aligned}$$

where ϵ is the probability that two functions will be the same. When $\left|\mathcal{F}_+\right| \cdot \epsilon$ is much larger than one, then this is well approximated by:

$$n \leq \sqrt{2\left|\mathcal{F}_+\right| \epsilon}. \qquad (16)$$

For example, problems as small as those with a 16-bit input and an 8-bit output have $|\mathcal{F}| \approx 2.6 \cdot 10^{157826}$. Assuming $n = 10^{12}$ and $\left|\mathcal{F}_+\right| = 10^{100}$ (hundreds of thousands of orders of magnitude smaller), (13) states the probability that any two samples will have the same function is approximately $5.0 \cdot 10^{-77}$. Thus, in sampling the problem domain to build $\hat{P}_+(f)$, it is extremely likely that a particular function will be sampled once or not at all. This leads to the two-valued nature of $\hat{P}_+(f)$ in practice. Equation (16) shows how many samples are permitted while maintaining the two-valued nature with approximately $1 - \epsilon$ confidence. Continuing the above example, when setting the probability of failure $\epsilon \approx e^{-100}$, we see n may be as large as $2.7 \cdot 10^{28}$.

■