WHY MACHINE LEARNING WORKS

George D. Montañez

May 2017

CMU-ML-17-100

Machine Learning Department
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA

Thesis Committee:
Cosma R. Shalizi, Chair
Roni Rosenfeld
Geoff Gordon
Milos Hauskrecht

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Copyright © 2017 George D. Montañez

This research was sponsored by the National Science Foundation under grant numbers DMS1207759, DMS1418124 and DGE1252522; a gift from the Microsoft Corporation; and a graduate fellowship from the Ford Foundation. Travel grants and one-time support provided by the IEEE Computational Intelligence Society, Center for Evolutionary Informatics, and Harvey Mudd College.

This thesis contains content from the following publications:

2017	Montañez G, Shalizi C, "Why Machine Learning Works, In One Equation" *(In Prep.)*
2017	Montañez G, Finley T, "Kernel Density Optimization" *(In Prep.)*
2016	Montañez G, "The Famine of Forte: Few Search Problems Greatly Favor Your Algorithm." *(In Prep.)*
2015	Montañez G, Amizadeh S, Laptev N, "Inertial Hidden Markov Models: Modeling Change in Multivariate Time Series." *AAAI Conference on Artificial Intelligence (AAAI-15)*, 2015.
2013	Montañez G, "Bounding the Number of Favorable Functions in Stochastic Search." In *Evolutionary Computation (CEC), 2013 IEEE Congress on*, pages 3019–3026, 2013.

Keywords: machine learning, algorithmic search, famine of forte, no free lunch, dependence

Dedicated to heroes everywhere, who stood their ground in the face of opposition and did what was right. To Jesus Christ, the first and last Hero, the iron pillar and bronze wall. "In this world you will have trouble. But take heart! I have overcome the world."

Abstract

To better understand why machine learning works, we cast learning problems as searches and characterize what makes searches successful. We prove that any search algorithm can only perform well on a narrow subset of problems, and show the effects of dependence on raising the probability of success for searches. We examine two popular ways of understanding what makes machine learning work, empirical risk minimization and compression, and show how they fit within our search framework. Leveraging the "dependence-first" view of learning, we apply this knowledge to areas of unsupervised time-series segmentation and automated hyperparameter optimization, developing new algorithms with strong empirical performance on real-world problem classes.

Contents

List of Figures

List of Tables

Part I

Introduction and Background

Chapter 1

Introduction

WHY does machine learning work? While wildly successful in business and science, machine learning has remained something of a mysterious black-box to all but a trained few. Even the initiated often have trouble articulating what exactly makes machine learning work in general. Although it is easy enough to point someone to a textbook on statistical learning theory, giving a concise (and correct) explanation becomes a little more difficult. Part of this thesis aims to help remedy the situation by offering a succinct yet clear answer to the question, based on two simple ideas that underlie most, and perhaps all, of machine learning: **search** and **dependence**. We reformulate several types of machine learning (e.g., classification, regression, density estimation, hyperparameter optimization) as search problems within a unified framework, formalizing a view of learning suggested by Mitchell [56]. Viewing machine learning as a search abstracts away many of the specific complexities of individual learning processes, allowing us to gain insight from a simplified view. Under this view, our original question can be transformed into two simpler ones:

1. What proportion of searches are successful?
2. For those that are, what makes them successful?

Having reduced machine learning to a type of search, if we can discover what makes generic searches successful then we also discover what makes machine learning successful.

Towards that end, we prove theorems regarding the probability of success for searches and explore what factors enable their success. We are introduced to No Free Lunch [71, 94, 95] and other Conservation of Information [22, 55, 55, 69] theorems, which demonstrate the need for correct exploitation of dependence to gain better-than-chance learning. As part of the process, we rederive and strengthen some existing historical results, and explore how our search framework compares and contrasts to other popular learning paradigms, such as empirical risk minimization and learning via compression (minimum description length). Lastly, we apply these insights to two concrete application areas, showing how dependence within those problems leads to the possibility of greater-than-chance success.

1.1 Contributions

A central contribution of this thesis is to develop a formal framework for characterizing search problems, into which we reduce several broad classes of machine learning problems (Chapters 3 and 4). Having established the link between machine learning and search, we demonstrate that favorable search problems must necessarily be rare (Chapter 5, Theorem 2). Our work departs from No Free Lunch results (namely, that the mean performance across sets of problems is fixed for all algorithms) to show that the proportion of favorable problems is strictly bounded in relation to the inherent problem difficulty and the degree of improvement sought (i.e., not just the mean performance is bounded). Our results continue to hold for sets of objective functions that are *not* closed-under-permutation, extending traditional No Free Lunch theorems. Furthermore, the bounds presented here do not depend on any distributional assumptions on the space of possible problems, such that the proportion of favorable problems is small regardless of which distribution holds over them in the real world. This directly answers critiques aimed at No Free Lunch results arguing against a uniform distribution on problems in the real world (cf. [64]), since given *any* distribution over possible problems, there are still relatively few favorable problems within the set one is taking the distribution over.

As a corollary, we prove the information costs of finding any favorable search problem is bounded below by the number of bits "advantage" gained by the algorithm on such problems. We do this by using an active information transform to measure performance improvement in bits [17], proving a conservation of information result [17, 22, 57] that shows the amount of information required to locate a search problem giving b bits of expected information is at least b bits. Thus, to get an expected net gain of information, the true distribution over search problems must be biased towards favorable problems for a given algorithm. This places a floor on the minimal information costs for finding favorable problems.

In the same chapter, we establish the equivalence between the expected per-query probability of success for an algorithm and the probability of a successful single-query search under some induced conditional distribution, which we call a *strategy*. Each algorithm maps to a strategy, and we prove an upper-bound on the proportion of favorable strategies for a fixed problem (Theorem 3). If finding a good search problem for a fixed algorithm is hard, then so is finding a good search strategy for a fixed problem. Thus, the matching of problems to algorithm strategies is provably difficult, regardless of which is fixed and which varies.

Given that any fixed algorithm can only perform well on a limited proportion of possible problems, we then turn our attention to the problems for which they do perform well, and ask what allows the algorithm to succeed in those circumstances. To that end, we characterize and bound the single-query probability of success for an algorithm based on information resources (Chapter 5, Theorems 4 and 5). Namely, we relate the degree of dependence (measured in mutual information) between target sets and external information resources, such as objective functions, inaccurate measurements or sets of training data, to the maximum improvement in search performance. We prove that for a fixed target-sparseness and given an algorithm $\mathcal{A}$, the single-query probability of success for the algorithm is bounded as

$$\Pr(X \in T; \mathcal{A}) \leq \frac{I(T; F) + D(P_T \| \mathcal{U}_T) + 1}{I_\Omega} \tag{1.1}$$

where $I(T;F)$ is the mutual information between target set T (as a random variable) and external information resource F, $D(P_T \| \mathcal{U}_T)$ is the Kullback-Leibler divergence between the marginal distribution on T and the uniform distribution on target sets, and I_Ω is the baseline information cost for the search problem due to sparseness. This simple equation takes into account degree of dependence, target sparseness, target function uncertainty, and the contribution of random luck. It is surprising that such well-known quantities appear in the course of simply trying to upper-bound the probability of success. We strengthen this in Theorem 5 to give a closed form solution to the single-query probability of success for an algorithm, again consisting of easily interpretable components.

Given our theoretical results and search framework, we are able to rederive an existing result from Culberson [15] as a corollary (Corollary 3) and formally prove (and extend) an early result from Mitchell [55] showing the need for inductive bias in classification (Theorems 6 and 7).

Chapter 6 explores the relationship of this work to established research in statistical learning theory, particularly in regards to empirical risk minimization. In Chapter 7 we discuss compression and its relation to generalization performance, including a discussion of the Minimum Description Length framework. We find that while a preference for simpler models can be a good strategy in some common situations, a preference for simplicity does not always translate into successful machine learning, disqualifying Occam's razor and related notions as a complete (i.e., necessary and sufficient) explanation for why machine learning works. Within the same chapter we prove a simple bound for the proportion of learnable binary concepts for deterministic classification algorithms with restricted data (Theorem 9) and give examples.

Moving from the theoretical world into the applied, Chapters 9 and 10 explore how making assumptions that enforce dependence (namely, temporal and spatial regularity) can lead to strong algorithms in the areas of multivariate time-series segmentation and hyperparameter optimization, respectively. Our "dependence-first" view of learning helps us to identify and exploit potential areas of dependence, leveraging them to develop novel applied machine learning algorithms with good empirical performance on real-world problems.

1.2 Answers of Increasing Complexity

In answer to our original question, we will offer the following one word, one sentence, and one paragraph answers. Why does machine learning work?

1. Dependence.
2. What is seen tells us something about what is unseen.
3. We are trying to recover an object based on observations; dependence within the problem controls how informative each observation can be, while biases (shaped by underlying assumptions) control how informative each observation actually is. Information theory then governs how fast we recover our object given the dependence, assumptions and biases.

Each of these answers will be justified during the course of the thesis.

1.3 Computational and Informational Complexity

It should be noted that this thesis views machine learning from a search and information theoretic perspective, focusing on the information constraints on learning problems. Additionally, there are *computational* constraints that must be dealt with in machine learning, focusing on scalability and engineering challenges. Here we focus solely on the informational challenges, while still acknowledging the importance of the computational aspects of machine learning; they are simply beyond the scope of this thesis.

Chapter 2

Related Work

2.1 Existing Frameworks

SINCE at least the time of Hume [38], the question of why induction is justified has been one of central importance in philosophy. Machine learning functions as a strategy of induction, generalizing from finite sets of examples to general truths, which are assumed to hold over even unseen examples. While Hume's problem of induction remains an open topic of philosophical debate [85], we will assume that induction in general is justified, and investigate what makes machine learning work as a specifically powerful type of induction.

Others have suggested general underlying mechanisms for machine learning's success, including Vapnik's VC theory [81], the Minimum Description Length principle [34] (which is a contemporary formalization of Occam's Razor), and Solomonoff induction [41, 78, 79]. We will explore how our search framework overlaps with some of these previous frameworks, as well as how it complements them, in Chapters 6 and 7.

2.2 Generalization as Search

The formal framework of machine learning as search presented here is inspired by work of Mitchell [56], which casts concept learning as a search process through a hypothesis (or *generalization*) space, where different learning methods correspond to different search algorithms within that space. Mitchell compares various methods of concept learning in terms of a single framework. His analysis is done under three assumptions:

1. The generalizations sought are those strictly consistent with the training data,
2. There are no errors in the labeling of training examples (no noise), and
3. The hypothesis space contains the true concept.

These assumptions are examined in later parts of the paper when contrasting the methods presented by the author to statistical learning methods, which allow for noise and represent training instances as points in d-dimensional space to be split by a hyperplane, and which Mitchell refers to as an "important subclass of generalization problems."

In Mitchell's framework, each hypothesis in the hypothesis space can be viewed as general or specific, depending on how many instances in the instance space are matched by the hypothesis. The *more-specific-than* relation between hypotheses, where $\{x : g_1(x) = 1\} \subseteq \{x : g_2(x) = 1\}$ indicates that g_1 is more specific than g_2, provides a partial ordering over the hypothesis space, allowing a structured search over that space. A few different search strategies are outlined, including a *version space* method, and their time and space complexities are provided.

Mitchell explores cases where there are not enough training examples to reduce the version space to a single hypothesis. In such cases, there exists a need for voting or some other method of choosing among the remaining hypotheses consistent with the training data. Methods of preferring one hypothesis to another with regard to anything other than strict consistency with the training data are examples of inductive bias in learning algorithms, as are choices made in the hypothesis space construction. Biases are encoded in the choice of "generalization language" (meaning the possible concepts that can be represented in the hypothesis space), and could be used to incorporate prior domain knowledge into the search problem. The problem of selecting "good" generalization languages (choosing a correct inductive bias) is highlighted as an important open problem in learning that was poorly understood at the time.

A connection is made to active learning (though not by that name), since the most informative labels for instances are those for which the current version space disagrees on with an almost equal number of hypotheses. In other words, those instances that are the most informative are those for which the current version space has the most trouble reliably classifying.

2.3 Constraints on Search: No Free Lunch and Conservation Theorems

There exists a substantial body of work proving conservation results in learning and search. That work has provided initial inspiration for many of the results in this manuscript, and we will review some of it here.

2.3.1 No Free Lunch Theorems

The canonical work on the subject was done by Wolpert in a series of papers [91, 92, 94, 95]. He formally proves what came to be known as *the* No Free Lunch (NFL) theorems for optimization, showing that uniformly averaged over all possible problems, every search algorithm has identical performance. This holds no matter which performance metric is chosen as long as it is based solely on the points sampled. Applying similar principles to supervised learning, Wolpert is able to show that when considering generalization error over a separate held-out set of data, no classification algorithm has superior performance to any other if averaging over all possible classification problems. Our work can be seen as a unification of some of Wolpert's work, allowing one to prove theorems for both optimization and supervised learning within the *same* framework.

2.3.2 Conservation of Generalization Performance

Schaffer [69] gave one of the earliest formal proofs of a conservation theorem for supervised learning. Schaffer shows that inductive inference is a zero-sum enterprise, with better-than-chance performance on any set of problems being offset exactly by worse-than-chance performance on the remaining set of problems. To characterize how good an algorithm is, we test it on a set of problems. The larger the set, the more confident our conclusions concerning the behavior of the algorithm. Schaffer shows that taking this process to the extreme, where all discrete problems of any fixed length are considered and our judgments concerning behavior should be most confident, we find that all learning algorithms have identical generalization performance.

In the problem setting considered by Schaffer, training and test data are allowed to be drawn from the same distribution, but instances that have already been seen during training are ignored during testing. This gives the off-training-set generalization error problem setting investigated by Wolpert [94] and others. Using his result, he shows that tweaking an algorithm to perform well on a larger set of problems means that you are simultaneously decreasing performance on all remaining problems. Thus, algorithm design is simply adjusting inductive bias to fit the problem(s) at hand.

Schaffer comments on a common attitude concerning No Free Lunch results, namely that "the conservation law is theoretically sound, but practically irrelevant." In his opinion, this view is common because each induction algorithm is only applied to a small subset of the space of all possible problems. Therefore, if bad generalization performance can be sequestered to the region outside this subset, then any constraints imposed by the generalization law can be safely ignored as having no practical consequence. Since we design learning algorithms to be used in the real-world, many hope that this is indeed the case. As an argument against this view, he highlights many instances in the literature where worse-than-chance learner behavior had been observed and reported on real-world problems, such as cases where avoidance of overfitting led to degraded generalization performance and where accuracy on an important real-world problem severely and continuously decreased from .85 to below .50 as a standard overfitting avoidance method was increasingly applied. Given the self-selection bias against including examples of negative (worse-than-chance) performance on tasks, it is remarkable that the examples were continually popping up. Most researchers view such results as anomalous, but according to Schaffer "the conservation law suggests instead that they ought to be expected, and that, the more we think to look for them, the more often examples will be identified in contexts of practical importance." Thus, even for real-world problem sets, we encounter cases where our algorithms work worse than random chance.

2.3.3 An Algorithmic View of No Free Lunch

Culberson [15] develops his own informal arguments for the No Free Lunch theorems, using adversary arguments from theoretical computer science to easily re-prove some of the statements of Wolpert and Macready. The goal of his paper is to relate NFL results to standard results in algorithmic theory, drawing connections between the two. For example, he argues that if $P \neq NP$, then no deterministic or randomized algorithm can solve arbitrary problems in the larger class in polynomial time, making statements in traditional complexity theory even more restrictive than the NFL theorems.

In the paper, evolutionary algorithms (EAs) are claimed to be “no prior knowledge” algorithms, which means EAs only gain knowledge of the problem through the external information object that is a fitness function. In proposing configurations to be evaluated, the environment is said to act as a *black box* and return an evaluation of that particular configuration. Culberson states that Wolpert and Macready’s NFL theorem for optimization prove that it is unreasonable to expect optimization algorithms to do well under these circumstances, since they show that all algorithms (including dumb random sampling) have identical averaged behavior when faced with such a black box environment.

He proves a version of the NFL theorem using an informal adversary argument, and gives a second NFL theorem in the spirit of Schaffer’s Conservation Law for Generalization Performance, namely, that “any algorithm that attempts to exploit some property of functions in a subclass of $\mathcal{F}$ will be subject to deception for some other class, in the sense that it will perform worse than random search.”

Most relevant to the results in this thesis, Culberson gives a theorem resembling the Famine of Forte for a much more restricted setting, showing that if we restrict ourselves to bijective functions from S^n (the space of all configurations, which we call Ω) to $\mathcal{R}$ (the finite range of values the function can take on when evaluated on strings in S^n), and if we use a non-repeating algorithm and consider the problem of selecting the optimum string within no more than an expected ξ queries, then that proportion is bounded above by $\xi/2^n$, for any fixed algorithm A where $|S^n| = 2^n$. This says that an algorithm can only select a target in polynomial expected time on an exponentially small fraction of the set of bijective functions, i.e., no more than $n^k/2^n$. Since he is dealing with the proportion of functions for which an algorithm can perform well in expectation, the similarity in spirit to the Famine of Forte result is clear, yet because his problem setting is also greatly restricted (i.e., considering only the closed-under-permutation set of bijective functions for non-repeating algorithms with a pre-specified single-item target set), his informal proof is much simpler than the proof given here for our more general result.

Culberson reasons that applying an algorithm to all possible functions (or any closed-under-permutation set of functions) gives the adversary too much power, in that it allows an adversary to return values that are completely independent of the algorithm in use. This is an early recognition of the importance of dependence for successful search. He evaluates issues in evolutionary algorithms, such as the implications of NFL for Holland’s once-celebrated Schema theorem [1, 36], and explores why the NFL results came as such a shock to the evolutionary computing community. Assuming that something like an evolutionary algorithm was responsible for the emergence and diversification of life, Culberson explores ways that evolution might escape some of the more damning implications of the NFL. For example, given that the universe is not as random as it could be, and that there is regularity and structure in physical laws, perhaps this restriction on possible problems is sufficient to escape application of NFL-type theorems. Yet, he points out that many NP-hard optimization problems remain NP-complete even when their complexity is restricted, such as determining whether a graph is 3-colorable even when restricting to planar graphs of degree at most 4, and thus does not seem to fully endorse this particular solution to the paradox. As he states in another part of the paper: “Although most researchers would readily accept that there must be some exploitable structure in a function if an algorithm is going to make progress, Wolpert and Macready make the further point that unless the operators of the algorithm are correlated to those features, there is no reason to believe the algorithm will do better than random search."

Ultimately, he argues that removing the assumption that life is optimizing may be crucial to resolving the issue and offers preliminary evidence in favor of that conclusion.

2.3.4 Necessary and Sufficient Conditions for No Free Lunch

Schumacher et al. [71] prove a sharpened No Free Lunch theorem, namely, that a set being closed-under-permutation (CUP) is a necessary and sufficient condition for the NFL to hold, when all functions in the set are equally likely. They define a *performance vector* $V(A, f)$ as an ordered set of fitness values for points sampled by a search algorithm A during its search on function f. They also define an *overall measure* as a function that maps a set of performance vectors to some real value. These overall measures can be used to compare the overall performance of two algorithms, which produce sets of performance vectors when applied to a set of functions $\mathcal{F}$. The problem setting is for deterministic algorithms that do not resample points in the search space.

The paper describes four equivalent statements of the NFL theorem:

- NFL1: For any overall measure, each algorithm performs equally well.
- NFL2: (Radcliff and Surry 1995) For any two algorithms A and B, and for any function f_1, there exists a function f_2 such that $V(A, f_1) = V(B, f_2)$.
- NFL3: (Schumacher 2000) Every algorithm generates precisely the same collection of performance vectors when all functions are considered.
- NFL4: (Wolpert and Macready 1995) For any equally weighted overall measure, each algorithm will perform equally well.

A *weighted overall measure* takes a performance vector measure M applied to each performance vector and weights it, namely $W(f)M(V(A, f))$ and sums over all f. Thus, the fourth statement NFL4 takes the special case when all weights are equally likely.

The authors stress that "[a]n even stronger consequence, which seems not to have been properly appreciated, is that *all* algorithms are equally specialized" (as all produce the same collection of performance vectors). Furthermore, it is pointed out that if *any* algorithm is robust and general purpose, then every algorithm is and if one is not robust and general purpose, then no algorithm can be.

Lastly, it is shown that NFL results hold regardless of the level of compression in the closed under permutation set of functions $\mathcal{F}$. For example, needle-in-a-haystack functions are highly compressible, but an NFL result still holds for any CUP set of those.

2.3.5 Few Sets of Functions are CUP

Igel and Touissant [42] explore whether or not NFL results are expected to hold over typical sets of functions, and present a few main results. First, given that NFL only holds for sets of functions that are closed under permutation, and since the fraction of subsets which are closed under permutation drops dramatically as the size of input and output spaces are increased, the NFL holds on almost no subsets of the set of all possible functions. If the set of all possible functions has $2^{|\mathcal{Y}|^{|\mathcal{X}|}} - 1$ non-empty subsets, Theorem 2 gives the number of subsets that are closed under permutation:

> **Theorem 2:** The number of non-empty subsets of $\mathcal{Y}^{\mathcal{X}}$ that are CUP is given by $2^{\binom{|\mathcal{X}+|\mathcal{Y}|-1}{|\mathcal{X}|}} - 1$,

where $\mathcal{X}$ is the input space, $\mathcal{Y}$ is the output space, and $\mathcal{Y}^{\mathcal{X}}$ is the set of all possible functions over $\mathcal{X}$ and $\mathcal{Y}$. Even for $|\mathcal{X}| = 8$ and $|\mathcal{Y}| = 2$, the fraction of sets that is CUP is less than 10^{-74}.

Second, the paper demonstrates that for subsets of functions for which non-trivial[1] neighborhood relations hold, the NFL does not hold. The reason for this is given in Theorem 3,

> **Theorem 3:** A non-trivial neighborhood on $\mathcal{X}$ is not invariant under permutations of $\mathcal{X}$.

They then prove corollaries of this theorem, such as showing that if we restrict the steepness of neighborhoods for a subset of functions (steepness being a measure of the range of values in a neighborhood) to be less than the maximum for that same subset (considering pairs on points not in a neighborhood, as well), then the subset is not CUP. They also show this is true if we restrict the number of local minima of a function to be less than the maximum possible.

As the conclusion states, the authors

> ...have shown that the statement "I'm only interested in a subset F of all possible functions, so the NFL theorems do not apply" is true with probability close to one... [and] the statements "In my application domain, functions with the maximum number of local minima are not realistic" and "For some components, the objective functions under consideration will not have the maximal possible steepness" lead to situations where NFL does not hold.

2.3.6 Focused No Free Lunch Theorems for non-CUP Sets

Whitley and Rowe [71] explore cases where two or more algorithms have identical performance over a non-closed under permutation set of functions. Such sets of functions are called *focused sets*, and may be much smaller than the full permutation closure.

It is a known result that if `PermClosure`(β) represents the permutation closure of set of functions β and algorithm A_1 has better performance over β than algorithm A_2, then A_2 must have better performance over `PermClosure`(β)$-\beta$, according to the No Free Lunch theorems. However, the authors note that β is usually small and `PermClosure`(β) $-$ β huge. Focused NFL theorems can show that an NFL result holds between A_1 and A_2 over set β and set $C(\beta)-\beta$, where $C(\beta)$ is a focused set and is small.

The authors discuss Gray versus Binary representations of an instance space, and prove that for the Gray vs. Binary algorithm problem setting, for *any* function f_b there exists a focused set of functions (a subset of the permutation closure) with less than $2L$ members, where L is the length of the representations for the Gray and Binary coded instances. This set will be much smaller than the full permutation closure, and the result holds for any pair of algorithms whose only difference is their choice of representation of the instance space.

[1]A neighborhood relation is non-trivial if it doesn't apply to all pairs of points, or to none.

Next, the authors show that the Sharpened No Free Lunch theorem is a special case of focused sets, and follows as a corollary from a proved lemma. They also provide a heuristic algorithm for discovering focused sets for a set of algorithms, but state that it may not always be possible to find a focused set any smaller than the full permutation closure.

Thus, when comparing the performance of some finite number of search algorithms over a set of functions, equal performance for members in that set can occur within a focused set much smaller than the full permutation enclosure of the functions, and an NFL-like result may hold even when the set of functions being evaluated is not closed under permutation.

2.3.7 Beyond No Free Lunch

Marshall and T. Hinton extend classical No Free Lunch results to the case where algorithms can revisit points, and show this breaks permutation closure, thus violating the NFL conditions [52]. They show that in the case of revisiting algorithms, under any 'sensible' performance measure (defined later in the paper), random enumeration of a space without revisiting has greater expected performance that a randomly selected revisiting algorithm.

To demonstrate that revisiting of points breaks permutation closure, they show that each revisiting algorithm over a CUP set of objective functions is equivalent to a non-revisiting algorithm over a larger search space under a set of objective functions that are no longer CUP. A corollary is stated that all "Free Lunches" can be expressed non-(block)uniform priors holding over problem sets that are closed under permutation. This follows from the fact that non-CUP sets are special cases of non-uniform distributions over CUP sets. Thus the distribution over the problem set is all that needs to be taken into consideration.

The authors consider 'sensible' performance measures, which are those that measure performance only in reference to co-domain values seen during a search where the following constraint holds $S_1 \subseteq S_2 \Rightarrow M(S_1) \leq M(S_2)$, S_1 and S_2 being the sets of co-domain values and $M(\cdot)$ being the measure. The proposition is then stated that random enumeration has better than or equal to performance for a randomly selected revisiting algorithm under any sensible performance measure. It is pointed out that random enumeration has excellent time and space complexity, and thus should be used as the baseline benchmark for algorithm comparisons.

The authors argue that just because we know that NFL conditions are violated (allowing some algorithm to possibly have better than random chance performance) this does not help us locate such an algorithm.

2.3.8 Continuous Lunches Are Free!

In traditional work on NFL theory, all domains and codomains remain finite. Auger and Teytaud [5] extend analysis to countably infinite and uncountable domains, and show that by defining NFL conditions in terms of measure theory, certain specific types of NFL behavior are impossible in uncountable domains, and only possible in certain situations for countably infinite domains. Given their results, they claim that continuous lunches are free and argue that,

> The deep reason for this fact [that continuous lunches are free] is that in "bigger" spaces (and continuous spaces are "very" big), random fields (and distributions of

fitness functions are non-trivial random fields in the continuous case) necessarily have correlations.

The fact that their results rely on special ways of defining random fitnesses and free lunches suggests that perhaps their results do not hold for all continuous spaces in general. Lockett and Miikkulainen argue that this is the case [51].

2.3.9 Continuous Lunches Are Not Free

Lockett and Miikkulainen [51] reformulate a probabilistic version of NFL and show that the previous results of Auger and Teytaud [5] do not hold in general, but rely on special conditions stated in their formulation. Without these restrictions one can obtain NFL theorems even in continuous spaces. In particular, their restriction to Lebesgue processes and requirement for proper medians is sufficient to ensure the exclusion of many measures for which continuous NFL holds. As Lockett and Miikkulainen state,

> Thus what Auger and Teytaud called a *random fitness* is likely to be a *Baire random fitness with a proper median*. The requirement of a proper median excludes most fitness measures. [...] Thus even if their proof holds, it does not mean that continuous NFL is impossible, but only that random fitness with the NFL property must not have a proper median nor be induced by a Lebesgue process. The definition of a proper median is too strict; it excludes perfectly reasonable fitness measures.

Example 3 of that paper (titled "Continuous Lunches Are Not Free") gives an example of the NFL for a continuous space, and the authors explicity state that Auger and Teytaud's general claim that continuous lunches are free is false. Thus, by careful use of measure theoretic definitions, Lockett and Miikkulainen show that the NFL continues to hold for even continuous spaces.

2.4 Bias in Learning

Another line of research relevant to the current thesis is the characterization of various types of bias within search and learning, and how they contribute to improved performance. We review a few key results here.

2.4.1 Futility of Bias-Free Learning

Mitchell [55] presents the futility of bias-free learning and highlights the importance of leveraging task specific domain knowledge to improve generalization accuracy. The paper defines inductive bias as any basis for choosing one hypothesis over another besides strict consistency with the observed training data. Two common sources of such inductive bias include restricting the hypothesis space (only certain concepts can be represented) or biasing the search procedure through a hypothesis space, effectively placing a (possibly probabilistic) ranking over the hypotheses in that space. Mitchell points out that an unbiased learner is one for which its inferences logically follow from the training instances; since classifications of new instances do not logically follow from the training data, the unbiased learner cannot make any (better than random guessing) claims concerning their class labels. Thus unbiased learning is futile and the power of a generalization system follows directly from the strength of its biases. This line of thinking foreshadows Schaffer and Wolpert's later statements that induction is only an expression of bias. The paper also discusses some possibly justifiable forms of inductive bias, such as using domain knowledge to reduce uncertainty within the hypothesis space. The claim is made that progress in machine learning depends on understanding, and justifying, various forms of bias. An appeal is made for researchers to make examination of the effect of bias in controlling learning as explicit (and rigorous) as research into how training instances affect learning has been in the past.

2.4.2 Overfitting Avoidance as Bias

In [70], Schaeffer investigates how even standard techniques within machine learning, such as the avoidance of overfitting through regularization and decision tree pruning, are simply a form of bias and are not inherently beneficial. He argues there are situations where such a bias is appropriate, and other situations where overfitting avoidance reduces predictive accuracy. The paper examines decision tree pruning as a case study, and demonstrates realistic situations where pruning reduces performance.

It is argued that if overfitting avoidance has proved beneficial in practice, this is due to the problems selected by researchers, not due to any general statistical properties of overfitting avoidance. Had other problems been chosen, one would be able to support the conclusion that overfitting helps predictive accuracy, not harms it. In the words of the author, "the important point for researchers is to credit observed success to the appropriate application of bias rather than to construe them as evidence in favor of the bias itself."

To demonstrate these points, Schaeffer compares a *sophisticated* method of decision tree learning, which uses tenfold cross-validation to decide whether or not to prune a tree, and a *naive* strategy, which never prunes. Experiments were carried out using an instance space consisting of vectors with five boolean attributes, $[A_1, A_2, A_3, A_4, A_5]$, and boolean class labels. Noise was applied to the class label of each instance, changing the label with some probability. Several experiments were carried out, and for each, one of the two strategies was superior.

When the relation to be learned was the simple relationship $C = A_1$, the sophisticated strategy (pruning) performed best. When the relationship was $C = \texttt{Parity}(A_1, A_2, A_3, A_4, A_5)$, the naive strategy outperformed the sophisticated strategy. When the relationship was $C = A_1 \vee (A_2 \wedge A_3)$, the sophisticated strategy again performed best. Paradoxically, when label noise was increased for the experiments, the benefit of overfitting avoidance decreased, contrary to expectation. (For *attribute* noise, however, overfitting avoidance did in fact appear to confer empirical benefit.)

In another experiment, boolean functions were selected uniformly at random to be learned by the decision trees, and the naive strategy (no pruning) performed best. This was likely due to the fact that most functions are algorithmically random or nearly so, and are thus complex. Since overfitting avoidance is a bias favoring simpler representations, in cases where the assumption of simplicity does not hold performance will suffer.

The paper covers the effect of representation language on the benefits of overfitting avoidance. First, a new attribute language was created such that $B_i = \texttt{Parity}(A_1, \dots, A_i)$ for $i = 1, \dots, 5$. Thus, the parity function for all five original attributes, namely $\texttt{Parity}(A_1, \dots, A_5)$, is equal to B_5, and learning $C = \texttt{Parity}(A_1, A_2, A_3, A_4, A_5)$ is equivalent to learning $C = B_5$.

In the B attribute language, the parity function is as simple as the $C = A_1$ function discussed earlier, and the sophisticated strategy has superior performance. The opposite holds when the parity function is represented using the A attribute representation. Thus, representation can have a large effect on the simplicity of a function and on the advantage of overfitting avoidance when learning that function. As Schaeffer notes, "An overfitting avoidance scheme that serves as a bias towards models that are considered simple under one representation is at the same time a bias towards models that would be considered complex under another representation."

Furthermore, sparse functions (those with few zeros or few ones) were more suited to use of overfitting avoidance, regardless of the representation language. This is contrasted with initially simple functions such as $C = A_1$, where half of the instances have labels of 1 and the other half are labeled 0. Under most representations, this function is complex, while under the original representation language it is simple.

A point is made that although researchers discuss using overfitting avoidance to separate noise from structure in the data, it is not possible to make such distinctions using the training data. If many hypotheses of varying complexity are equally consistent with the training data, then nothing in the observed data can distinguish between them. In those cases, we must use domain knowledge to guide our choice. This follows as a consequence of Bayes' rule, where $P(h|d) \propto P(d|h)P(h)$, so that when the data are explained equally well by all models (i.e., $P(d|h)$ is the same for all h), the only possible way to increase the likelihood of one hypothesis over another is to have differences in $P(h)$, which are independent of the data.

The paper discusses some issues with specific applications of Occam's Razor and minimum description length, showing that using either of these is simply applying a bias which may or may not be applicable in certain problem domains. Biases are neither inherently good nor bad, only more or less appropriate. Schaeffer comments that while many data sets used in machine learning research do apparently have simple underlying relationships, this only tells us what types of problems researchers tend to work on. By itself, this is not evidence that problems in the real world tend to have simple underlying relationships.

2.4.3 Evaluating "Overfitting Avoidance as Bias"

Wolpert reviews [93] some of the results from Cullen Schaffer's previously mentioned paper [70], and concludes that there is theoretical justification for many of the claims presented there. He does this by showing they follow from the No Free Lunch theorems for supervised learning.

Of particular interest, he discusses Schaffer's experiment where boolean functions are selected uniformly at random and the naive strategy is found to perform better than the sophisticated strategy in the majority of cases examined. He notes that while the NFL states no learning algorithm will have better off-training-set performance when a uniform distribution over problems holds, Schaffer actually considered the conventional generalization error, which allows for overlap between training and testing data. Therefore, it could be the case that the naive strategy (no pruning) is simply better at reproducing the training set examples than the sophisticated strategy is.

Wolpert also considers the case when the size of the instance space grows much larger than the size of the training set, so that the conventional i.i.d. error converges to the off-training-set error in the limit. In these scenarios, he argues no learning algorithm has strictly superior generalization performance compared to any other, even when considering the conventional generalization error function, as within the PAC learning framework. He suggests caution when interpreting PAC learning results that may seem to indicate otherwise.

Lastly, Wolpert argues that while it is true that for some algorithms there exists a $P(f)$ for which it beats all other algorithms, there are some algorithms that are not optimal for any $P(f)$. Therefore, we can prefer algorithms that are optimal under some distribution to those that are not under any.

2.4.4 The Need for Specific Biases

Wilson and Martinez [90] make a case for the selection of *specific* biases in machine learning, which are narrow biases that only apply to the problem domain at hand. Since there is no inductive bias that is strictly better than any other when summed over all possible problems, the goal of machine learning becomes selecting a good bias for a given situation. However, they note that selecting a good bias is not trivial and is an area of much research in the machine learning community.

Wilson and Martinez propose the key to good generalization is thus to have a collection of powerful (specific) biases available, and to use domain knowledge to select from among these biases for a specific problem domain. Training examples alone cannot help with the selection of bias, other than eliminating hypotheses that disagree with the labels of the training instances. They contrast *theoretical average accuracy*, which is off-training-set generalization accuracy treating all functions as equally likely (i.e., under a uniform prior), with *practical average accuracy*, which allows for a nonuniform prior over problem functions. While all algorithms have identical theoretical average accuracy, algorithms can have differing practical average accuracy.

Discussing the role of domain knowledge in learning, they highlight the purpose of inductive bias, which is to allow for generalization to unseen instances. They introduce the notion of a meta-bias, which is needed to select among different inductive biases. Thus, inductive learning is reduced to selecting a good inductive bias for a particular problem domain, which is another learning problem. They do not explore whether learning a good inductive bias is quantitatively any easier than learning the original concept, which is an important question.

2.5 Improvements on Prior Work

This thesis offers some improvements over prior work in NFL and conservation of information. First, a unified search framework is developed which allows for addressing problems is search and learning simultaneously, while being flexible in the amount of agreement between the objective function values sampled and the true targets. Our framework will allow us to quantify the amount of dependence information between the targets and external information resources, giving us a method of bounding the probability of success as a function of dependence.

Second, our results hold for non-CUP sets of objective functions and for possibly repeating algorithms (of which non-repeating algorithms are a special subset). This makes our theoretical work apply more broadly, since we can address non-repeating algorithms and CUP sets of functions as special cases within our general framework.

Third, the conservation of information theorems proven in this thesis make progress towards answering the open question of whether selecting a meta-bias is any easier than selecting a good bias, showing that within the search framework, good biasing distributions (strategies) are just as rare as good elements, which can be viewed as preliminary evidence that the problem does not get easier as one moves up the bias hierarchy.

Part II

Theory

Chapter 3

An Algorithmic Search Framework[1]

WE cast machine learning as a type of search, where information from observations is exploited to uncover an item of interest (such as a parameter, a binary concept, or an underlying regression function). In this chapter we formally define our search framework and discuss how machine learning problems can be represented within it. Chapter 4 gives specific examples of such transformations for different types of learning problems.

3.1 The Search Problem

The search space, denoted Ω, contains the elements to be examined. We limit ourselves to finite, discrete search spaces, which entails little loss of generality when considering search spaces fully representable on physical computer hardware within a finite time. Let the **target set** $T \subseteq \Omega$ be a nonempty subset of the search space. The set T can be represented using a binary vector the size of Ω, where an indexed position evaluates to 1 whenever the corresponding element is in T and 0 otherwise. Thus, each T corresponds to exactly one binary vector of length $|\Omega|$, and vice versa. We refer to this one-to-one mapped binary vector as a **target function** and use the terms target function and target set interchangeably, depending on context. These target sets/functions will help us define our space of possible search problems, as we will see shortly.

In machine learning, elements from the search space Ω are evaluated according to some **external information resource**, such as a dataset. We abstract this resource as simply a finite length bit string, which could represent an objective function, a set of training data, or anything else. The resource can exist in coded form, and we make no assumption about the shape of this resource or its encoding. Our only requirement is that it can be used as an oracle, given an element from Ω or the null element. Specifically, we require the existence of two methods, one for initialization (given the null element) and one for evaluating queried points in Ω. We assume both methods are fully specified by the problem domain and external information resource. With a slight abuse of notation, we define an **information resource evaluation** as $F(\omega) := g(F, \omega)$, where g is an extraction function applied to the information resource and $\omega \in \Omega \cup \{\emptyset\}$. Therefore,

[1]This chapter reproduces content from Montañez, "The Famine of Forte: Few Search Problems Greatly Favor Your Algorithm" (arXiv 2016)

$F(\emptyset)$ represents the method used to extract initial information for the search algorithm (absent of any query), and $F(\omega)$ represents the evaluation of point ω under resource F.

A **search problem** is defined as a 3-tuple, (Ω, T, F), consisting of a search space, a target subset of the space and an external information resource F, respectively. Since the true target locations are hidden, any information gained by the search concerning the target locations is mediated through the external information resource F alone. Thus, the space of possible search problems includes many deceptive search problems, where the external resource provides misleading information about target locations, and many noisy problems. In the fully general case, there can be any relationship between T and F. Because we consider any and all degrees of dependence between external information resources and target locations, this effectively creates independence when considering the set as a whole, allowing some of our results to follow as a consequence.

However, in many natural settings, target locations and external information resources are tightly coupled. For example, we typically threshold objective function values to designate the target elements as those that meet or exceed some minimum. Doing so enforces dependence between target locations and objective functions, where the former is fully determined by the latter, once the threshold is known. This dependence causes direct correlation between the objective function (which is observable) and the target locations (which are not directly observable). We will demonstrate such correlation is exploitable, affecting the upper bound on the expected probability of success.

3.2 The Search Algorithm

An algorithmic search is any process that chooses elements of a search space to examine, given some history of elements already examined and information resource evaluations. The history consists of two parts: a query trace and a resource evaluation trace. A **query trace** is a record of points queried, indexed by time. A **resource evaluation trace** is a record of partial information extracted from F, also indexed by time. The information resource evaluations can be used to build elaborate predictive models (as is the case of Bayesian optimization methods), or ignored completely (as is done with uniform random sampling). The result is a time-indexed probability distribution P_i over the search space, used by the algorithm to sample the next element. The algorithm's internal state is updated at each time step according to the rules of the algorithm, resulting in an updated P_i, from which new elements are sampled and evaluated against the external information resource. The search is thus an iterative process, with the history h at time i represented as $h_i = (\omega, F(\omega))$. The problem domain defines how much initial information from F is given by $F(\emptyset)$, as well as how much (and what) information is extracted from F at each query.

We allow for both deterministic and randomized algorithms, since deterministic algorithms are equivalent to randomized algorithms with degenerate probability functions (i.e., they place all probability mass on a single point). Furthermore, any population based method can also be represented in this framework, by holding P_i fixed when selecting the m elements in the population, then considering the history (possibly limiting the horizon of the history to only m steps, creating a Markovian dependence on the previous population) and updating the probability

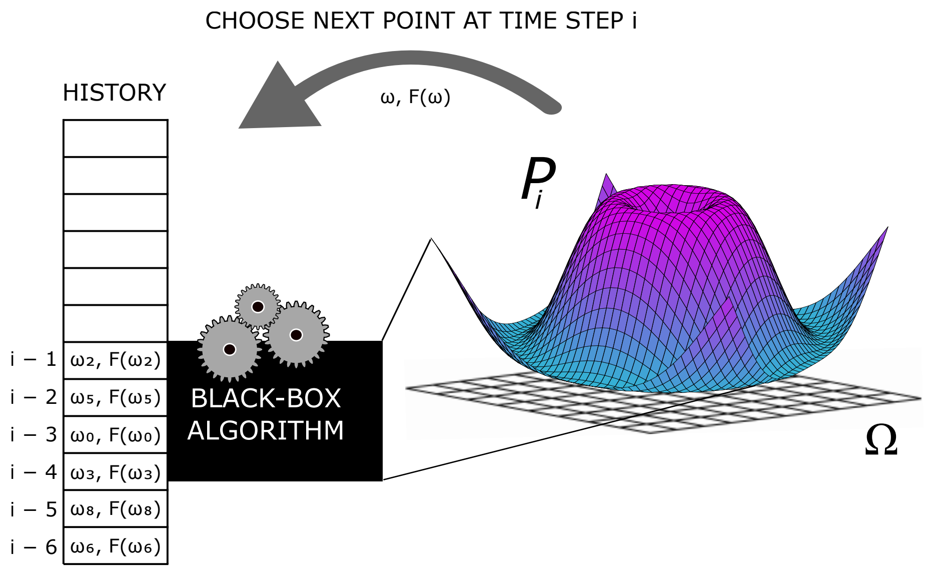


Figure 3.1: Black-box search algorithm. At time i the algorithm computes a probability distribution P_i over the search space Ω, using information from the history, and a new point is drawn according to P_i. The point is evaluated using external information resource F. The tuple $(\omega, F(\omega))$ is then added to the history at position i.

distribution over the search space for the next m steps.

Abstracting away the details of how such a distribution P_i is chosen, we can treat the search algorithm as a black box that somehow chooses elements of a search space to evaluate. A search is successful if an element in the target set is located during the search. Algorithm 1 outlines the steps followed by the black-box search algorithm and Figure 3.1 visually demonstrates the process.

Algorithm 1 Black-box Search Algorithm

```
1: Initialize $h_0 \leftarrow (\emptyset, F(\emptyset))$.
2: **for all** $i = 1, \dots, i_{\max}$ **do**
3:     Using history $h_{0:i-1}$, compute $P_i$, the distribution over $\Omega$.
4:     Sample element $\omega$ according to $P_i$.
5:     Set $h_i \leftarrow (\omega, F(\omega))$.
6: **end for**
7: **if** an element of $T$ is contained in any tuple of $h$ **then**
8:     Return *success*.
9: **else**
10:    Return *failure*.
11: **end if**
```

3.3 Measuring Performance

Since general search algorithms may vary the total number of sampling steps performed, we measure performance using the expected per-query probability of success,

$$q(T, F) = \mathbb{E}_{\tilde{P},H}\left[\frac{1}{|\tilde{P}|}\sum_{i=1}^{|\tilde{P}|} P_i(\omega \in T) \;\middle|\; F\right], \tag{3.1}$$

where $\tilde{P}$ is the sequence of probability distributions for sampling elements (with one distribution P_i for each time step i), H is the search history, and the T and F make the dependence on the search problem explicit. $|\tilde{P}|$ denotes the length of the sequence $\tilde{P}$, which equals the number of queries taken. The expectation is taken over all sources of randomness, which includes randomness over possible search histories and any randomness in constructing the various P_i from $h_{0:i-1}$ (if such a construction is not entirely deterministic). Taking the expectation over all sources of randomness is equivalent to measuring performance for samples drawn from an appropriately averaged distribution $\overline{P}(\cdot \mid F)$ (see Lemma 3). Because we are sampling from a fixed (after conditioning and expectation) probability distribution, the expected per-query probability of success for the algorithm is equivalent to the induced amount of probability mass allocated to target elements. Thus, each $\overline{P}(\cdot \mid F)$ demarks an equivalence class of search algorithms mapping to the same averaged distribution; we refer to these equivalence classes as **search strategies**.

We use uniform random sampling with replacement as our baseline search algorithm, which is a simple, always available strategy and we define $p(T, F)$ as the per-query probability of success for that method. In a natural way, $p(T, F)$ is a measure of the intrinsic difficulty of a search problem [17], absent of any side-information. The ratio $p(T, F)/q(T, F)$ quantifies the improvement achieved by a search algorithm over simple random sampling. Like an error quantity, when this ratio is small (i.e., less than one) the performance of the algorithm is better than uniform random sampling, but when it is larger than one, performance is worse. We will often write $p(T, F)$ simply as p, when the target set and information resource are clear from the context.

3.4 Notation

Table 3.1 gives the symbols used in this manuscript with their meanings.

Table 3.1: Notation

Symbol	Definition
Ω	Search space
ω	Element of search space Ω
$\mathcal{A}$	Search algorithm
T	Target set, $T \subseteq \Omega$
F	External information resource
$F(\emptyset)$	Initialization information
$F(\omega)$	Query feedback information
τ	Set of target sets
$\mathcal{B}_m$	Set of external information resources
$\tilde{P}$	Sequence of probability distributions on Ω
H	Search history
P_i	Probability distribution on Ω, $P_i \in \tilde{P}$
p	Baseline per-query probability of success under uniform random sampling: $p = \lvert T\rvert/\lvert\Omega\rvert$
q	$\mathbb{E}_{T,F}[q(T,F)]$
$q(T,F)$	Expected per-query probability of success, i.e. $\mathbb{E}_{\tilde{P},H}\left[\frac{1}{\lvert\tilde{P}\rvert}\sum_{i=1}^{\lvert\tilde{P}\rvert} P_i(\omega \in T) \mid F\right]$
$\overline{P}(\cdot \mid F)$	Averaged conditional distribution on Ω for an algorithm, also called a *search strategy*
$I_{q(T,F)}$	$-\log p/q(T,F)$
$\mathcal{L}$	Loss function
Ξ	Error functional
D	Training dataset
V	Test dataset
$\mu(A)$	measure on object A
$\mathcal{U}(A)$	uniform measure / distribution on object A

Chapter 4

Machine Learning as Search

IN this chapter we demonstrate how to reduce several types of learning problems to search problems within our formal framework, including the abstract learning problem considered by Vapnik [81] in his work on empirical risk minimization. Extensions to other types of learning problems not considered here should be equally straightforward.

4.1 Regression as Search

A central task for machine learning is *regression*, where a function from a set of inputs to a real-valued output is learned. In simple cases, the function g takes as input a real-valued variable and outputs a real-valued response, so $g : \mathbb{R} \to \mathbb{R}$. The functions can be more general than that, possibly taking multiple inputs and producing vector-valued outputs, for example. In machine learning, a set of data points are given along with their function evaluation values, and one must often recover a function $\widehat{g}$ that produces the evaluations at those data points with minimum (possibly penalized) loss.

Let

$$D = \{(x_1, y_1), \ldots, (x_n, y_n) : x_i \in \mathcal{X}, y_i \in \mathcal{Y} \text{ for } i = 1, \ldots, n\} \tag{4.1}$$

be a set of training data, where $\mathcal{X}, \mathcal{Y}$ are finite sets, and let

$$V = \{(x_1, y_1), \ldots, (x_m, y_m) : x_i \in \mathcal{X}, y_i \in \mathcal{Y} \text{ for } i = 1, \ldots, m\} \tag{4.2}$$

be a set of test data. Assume a learning algorithm $\mathcal{A}$ is allowed to choose from a large (but finite) collection of possible regression functions, Ω, and that it chooses the function $\widehat{g} \in \Omega$ according to loss function $\mathcal{L}$ and the training data D. Thus, $\widehat{g} = \mathcal{A}(\mathcal{L}, D)$, where the assignment may be stochastic.

Given Ω, a natural target set $T \subseteq \Omega$ is the set of all regression functions with acceptable performance when evaluated on the test set V, namely

$$T = \{g : g \in \Omega, \Xi(\mathcal{L}, g, V) \leq \epsilon\}, \tag{4.3}$$

where Ξ is an error functional operating on $\mathcal{L}$, regression function g and test set V, and ϵ is a scalar threshold. For the agnostic learning case (when it is not guaranteed that your class of functions contains a good function), Ξ can be made relative to the within-class optimum, as a regret. Also note that T is typically random, being a function of random set V.

The external information resource F consists of loss function $\mathcal{L}$ together with D, encoded in binary format. The ability to represent F as a finite bit string follows directly from the finiteness of $\mathcal{X}, \mathcal{Y}$ and the assumption that the loss function can be encoded as a finite binary procedure (such as a C++ method). F is also random whenever D is itself a random set. Dependence between D and V (via shared distributional assumptions, for example) creates dependence between F and T.

Given the elements (Ω, T, F) (which require $\mathcal{L}$, D and V), we can view each regression algorithm as a search that outputs a $\widehat{g} \in \Omega$ given external information resource F. Thus, regression problems can be represented naturally within our search framework.

4.2 Classification as Search

Classification can be viewed as a simple type of regression, where the output values are restricted to an often small set, such as $\{-1, 1\}$. The search space Ω is a finite set classification hypotheses, while a loss function such as zero-one loss is used to estimate the quality of any particular hypothesis within that space. We continue to have a dataset D, which consists of training instances and their class labels, and a test set V. In our framework, the primary difference between the general regression case and classification is that class labels are typically categorical, in that labels are not expected to have geometrical relationships corresponding to their cardinalities. In other words, just because labels 3 and 4 are closer to each other than labels 3 and 10 does not necessarily mean that class 3 is more similar to class 4 than it is to class 10. In contrast, regression output values do have inherent geometric relationships.

With that difference in mind, classification problems reduce to specific kinds of regression problems, and therefore can also be represented within our framework.

4.3 Clustering as Search

Given a sample of some points in a vector space, we often desire to cluster the points into distinct groups. Sometimes the number of groups is given beforehand, such as in the popular k-means++ clustering algorithm [4], and sometimes the algorithm determines the number of clusters itself, as in DBSCAN [23]. In both cases, the points are assigned to groups, partitioning the points into k distinct sets. Consider Figure 4.1, which is a set of one hundred points in two dimensional euclidean space. Visually, it may appear that all the points belong to a single cluster, or perhaps using a density-based algorithm with small neighborhood size, there may be several clusters. There is no universally agreed upon definition of what constitutes a "good" cluster, so researchers are forced to define what is a successful clustering for each task.

We will abstract the problem of clustering, showing how to represent it as a search problem within our general framework. At the highest level, there are basic elements for any clustering task:

1. A set of points in a vector space;

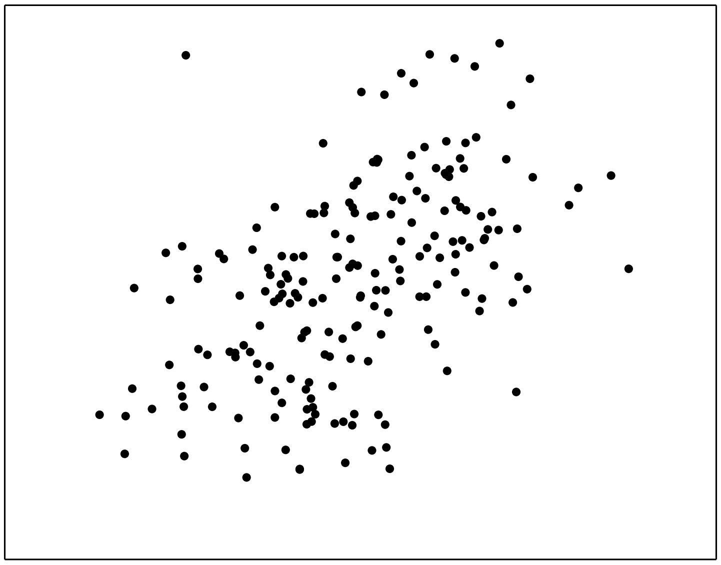


Figure 4.1: Points to be clustered in a vector space.

2. A fixed set of usually unknown clusters (more precisely, distributions), from which the points are drawn; and
3. A ground truth assignment of points to clusters (namely, the actual generation history of points).

The set of points are represented in a vector space, typically $\mathbb{R}^d$, and we want to recover the latent cluster assignments for all points. For *soft-clustering* algorithms, each point may be have membership in more than one cluster, whereas *hard-clustering* algorithms assign each point to at most one cluster. We will make a simplifying assumption that the number of clusters does not exceed the number of points and that the number of points, N, is finite. Given these assumptions, one can represent any clustering of the points among K classes as a real-valued $N \times K$ matrix, W, where entry W_{ij} represents the membership of point i in cluster j. For soft-clustering algorithms, these membership weights can take on real values, whereas hard-clustering algorithms produce binary values at all entries. Without loss of generality, assume all membership weights are normalized so that $0 \leq W_{ij} \leq 1$. As a further simplification, since $K \leq N$, we can represent W as simply an $N \times N$ matrix, where columns with all zeros correspond to non-existent clusters. In other words, if our algorithm produces only three clusters, then all but three columns of W will be equal to zero vectors.

The points themselves, being embedded in some vector space, have a representation within that space. The could be a vector of values in $\mathbb{R}^d$, or some other representation. Let us make a distinction between the points themselves and their representation within a particular vector space; thus a point can have many representations, given different vector spaces.

To cast any problem within our framework, we need to define our search space Ω, an external information resource F, and a target set T, and we assume the clustering algorithm $\mathcal{A}$ is fixed and given and the set of N points is also fixed and given. The search space Ω is the space of all possible assignment matrices W, and we discretize the search space by requiring finite precision representation of the assignment weights. The true assignments is represented by matrix W^*, and the target set T consists of all W such that $\mathcal{L}(W, W^*) \leq \epsilon$ for loss function $\mathcal{L}$ and threshold ϵ, namely

$$T = \{W : W \in \Omega, \mathcal{L}(W, W^*) \leq \epsilon\}. \tag{4.4}$$

Lastly, the external information resource F is a binary-coded vector space representation of all N points, where the values are again finite precision. We can assume that $F(\emptyset)$ returns the entire set of points, and that the algorithm produces at most one W matrix (acting as a single-query algorithm). Thus, we can fully represent general clustering problems within our framework.

To gain intuition as to how specific clustering tasks would look within our framework, let us return to the dataset presented in Figure 4.1. Although the point visually look like they belong to a single cluster, they were generated by sampling two independent sets of points i.i.d. from two multivariate Gaussian distributions. Figure 4.2 shows the ground-truth cluster assignments for this example. Given an unlabeled set of points like Figure 4.1 (and perhaps the true number of clusters), our algorithm must produce a W matrix that reliably separates the points into their distinctive groups. (We assume that our loss function takes into account label permutations, which are equivalent to transpositions of the columns of the W matrix.)

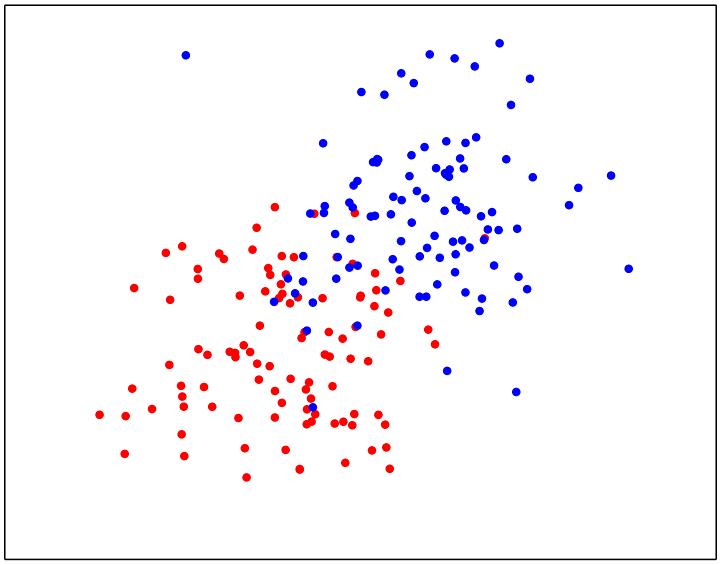


Figure 4.2: Samples drawn from two multivariate Gaussian distributions.

What becomes immediately apparent is the importance of the vector space representation of the points, as well as the set of points themselves. Had the points been projected to a different space that superimposed the two clusters, we would have little hope of recovering the true assignments. Thus, representations can be more (or less) informative of the true cluster assignments, and thus, provide more (or less) information for recovering the true W^*. This is similar to a case where you want to sort a group of people into `Republican` and `Democrat` clusters, but you have a choice in what attributes and features to use. If you represent the people using a bad feature representation (such as using gender-normalized shoe size, shirt color, number of eyes, number of vowels in first name), you have little chance of reliably clustering your population into their respective groups. However, if you use features highly correlated with the latent classes, such as voter registration party and zip code, you have a much higher probability of producing a clustering close to the truth. A good representation will increase the distance between classes, as in Figure 4.3, while a poor one will decrease it. Thus, we see the importance of dependence between the external information resource F, which is the chosen representation of a sampled set of points, and T, the target set of acceptable clusterings.

In addition to this, it should be clear that the way the points themselves are chosen, in *any* vector space representation, will also play a major role in the recovery of the true clusters. Choosing points in a noisy or adversarial manner instead of by sampling i.i.d. from the true underlying distributions will also decrease the information the set of points provides concerning the true cluster assignments.

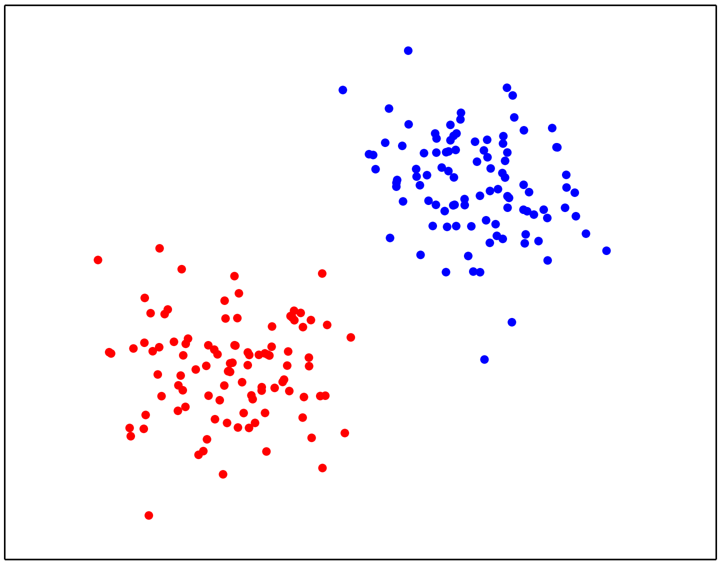


Figure 4.3: Samples drawn from two multivariate Gaussian distributions, with better separation among classes.

$$\theta \longrightarrow \boxed{\text{Density / Sampling Process}} \longrightarrow D$$

Figure 4.4: Parameter estimation

4.4 Parameter Estimation as Search

For parameter estimation, we begin with a large (but finite) search space Ω of possible vectors θ, each of which could be used as the input to a black-box process. The process, taking θ as input, then produces a set of outputs, denoted D. The task of parameter estimation is to use D to reason back to the correct θ that was used by the process.

The external information resource F consists of the output data D, suitably represented. The target set T can defined using a loss function $\mathcal{L}$ and threshold ϵ, namely

$$T = \{\theta' : \theta' \in \Omega, \mathcal{L}(\theta', \theta) \leq \epsilon\}. \tag{4.5}$$

This includes the true parameter vector θ, as well as any acceptably close alternatives. Restrictions detailing how D is produced in regards to θ (e.g., by sampling i.i.d. from a density parameterized by θ) control how F tends to vary for different θ, and consequently, how informative F is concerning θ.

4.5 Hyperparameter Optimization as Search

Many hyperparameter optimization methods lend themselves naturally to our search framework. Let Λ represent the hyperparameter space (appropriately discretized) and let $\phi(\lambda)$ denote the empirical metric used to measure how well the algorithm performs on the problem using hyperparameter configuration λ and let $\phi(\lambda^*)$ denote the best performance achievable for any configuration in Λ. For $\epsilon > 0$, we represent the corresponding search problem as follows:

- $\Omega = \Lambda$;
- $T = \{\lambda : \lambda \in \Lambda, |\phi(\lambda) - \phi(\lambda^*)| < \epsilon\}$;
- $F = \{\phi(\lambda_1), \dots, \phi(\lambda_{|\Omega|}))\}$;
- $F(\emptyset) = \emptyset;$ and
- $F(\lambda_i) = \phi(\lambda_i)$.

The sequential optimization procedure samples elements from the search space, evaluates them according to F, and adds the feedback to the history. Using the information in the history, it then chooses which element of the search space to sample next, as in Algorithm 1.

4.6 General Learning Problems as Search

Vapnik presents a generalization of learning that applies to classification, regression, and density estimation [81], which we can translate into our framework. Following Vapnik, let $P(z)$ be defined on space Z, and consider the parameterized set of functions $Q_\alpha(z), \alpha \in \Lambda$. The goal is to minimize $R(\alpha) = \int Q_\alpha(z)dP(z)$ for $\alpha \in \Lambda$, when $P(z)$ is unknown but an i.i.d. sample $z_1, \ldots, z_\ell$ is given. Let $R_{emp}(\alpha) = \frac{1}{\ell}\sum_{i=1}^{\ell} Q_\alpha(z_i)$ be the empirical risk.

To reduce this general problem to a search problem within our framework, assume Λ is finite, choose $\epsilon \in \mathbb{R}_{\geq 0}$, and let

- $\Omega = \Lambda$;
- $T = \{\alpha : \alpha \in \Lambda, R(\alpha) - \min_{\alpha' \in \Lambda} R(\alpha') < \epsilon\}$;
- $F = \{z_1, \ldots, z_\ell\}$;
- $F(\emptyset) = \{z_1, \ldots, z_\ell\};$ and
- $F(\alpha) = R_{emp}(\alpha)$.

Thus, any finite problem representable in Vapnik's learning framework is also directly representable within our search framework. The results presented here apply to all such problems considered in statistical learning theory.

Chapter 5

Theoretical Results[1]

5.1 Lemmata

We begin by proving several lemmata which will help us in establishing our main theorems.

Lemma 1 (Sauer-Shelah Inequality)**.** *For* $d \leq n$, $\sum_{j=0}^{d} \binom{n}{j} \leq \left(\frac{en}{d}\right)^d$.

Proof. We reproduce a simple proof of the Sauer-Shelah inequality [68] for completeness.

$$\sum_{j=0}^{d} \binom{n}{j} \leq \left(\frac{n}{d}\right)^d \sum_{j=0}^{d} \binom{n}{j} \left(\frac{d}{n}\right)^j \tag{5.1}$$

$$\leq \left(\frac{n}{d}\right)^d \sum_{j=0}^{n} \binom{n}{j} \left(\frac{d}{n}\right)^j \tag{5.2}$$

$$= \left(\frac{n}{d}\right)^d \left(1 + \frac{d}{n}\right)^n \tag{5.3}$$

$$\leq \left(\frac{n}{d}\right)^d \lim_{n\to\infty} \left(1 + \frac{d}{n}\right)^n \tag{5.4}$$

$$= \left(\frac{en}{d}\right)^d. \tag{5.5}$$

□

Lemma 2. $\sum_{j=0}^{\lfloor \frac{n}{2^b} \rfloor} \binom{n}{j} \leq 2^{n-b}$ *for* $b \geq 3$ *and* $n \geq 2^b$.

Proof. By the condition $b \geq 3$, we have

$$2b + \log_2 e \leq 2^b \tag{5.6}$$

[1]This chapter reproduces content from Montañez, "The Famine of Forte: Few Search Problems Greatly Favor Your Algorithm" (arXiv 2016)

which implies $2^{-b}(2b + \log_2 e) \leq 1$. Therefore,

$$\begin{aligned} 1 &\geq 2^{-b}(2b + \log_2 e) && (5.7)\\ &= \frac{b}{2^b} + \frac{b + \log_2 e}{2^b} && (5.8)\\ &\geq \frac{b}{n} + \frac{b + \log_2 e}{2^b}, && (5.9) \end{aligned}$$

using the condition $n \geq 2^b$, which implies

$$n \geq b + \frac{n}{2^b}(b + \log_2 e). \tag{5.10}$$

Thus,

$$\begin{aligned} 2^n &\geq 2^{b+\frac{n}{2^b}(b+\log_2 e)}\\ &= 2^b 2^{\frac{n}{2^b}(b+\log_2 e)}\\ &= 2^b \left(2^b 2^{\log_2 e}\right)^{\frac{n}{2^b}}\\ &= 2^b \left(2^b e\right)^{\frac{n}{2^b}}\\ &= 2^b \left(\frac{en}{\frac{n}{2^b}}\right)^{\frac{n}{2^b}}\\ &\geq 2^b \sum_{j=0}^{\frac{n}{2^b}} \binom{n}{j}\\ &\geq 2^b \sum_{j=0}^{\lfloor \frac{n}{2^b} \rfloor} \binom{n}{j}, \end{aligned}$$

where the penultimate inequality follows from the Sauer-Shelah inequality [68]. Dividing through by 2^b gives the desired result. □

Lemma 3. *(Expected Per Query Performance From Expected Distribution) Let t be a target set, $q(t, f)$ the expected per-query probability of success for an algorithm and ν be the conditional joint measure induced by that algorithm over finite sequences of probability distributions and search histories, conditioned on external information resource f. Denote a probability distribution sequence by $\tilde{P}$ and a search history by h. Let $\mathcal{U}(\tilde{P})$ denote a uniform distribution on elements of $\tilde{P}$ and define $\overline{P}(x \mid f) = \int \mathbb{E}_{P\sim\mathcal{U}(\tilde{P})}[P(x)]d\nu(\tilde{P}, h \mid f)$. Then,*

$$q(t, f) = \overline{P}(X \in t|f)$$

where $\overline{P}(X|f)$ is a probability distribution on the search space.

Proof. Begin by expanding the definition of $\mathbb{E}_{P\sim\mathcal{U}(\tilde{P})}[P(x)]$, being the average probability mass on element x under sequence $\tilde{P}$:

$$\mathbb{E}_{P\sim\mathcal{U}(\tilde{P})}[P(x)] = \frac{1}{|\tilde{P}|} \sum_{i=1}^{|\tilde{P}|} P_i(x).$$

Next, we confirm that $\overline{P}(x|f)$ is a proper probability distribution:

1. $\overline{P}(x|f) \geq 0$, being the integral of a nonnegative function;
2. $\overline{P}(x|f) \leq 1$, since

$$\overline{P}(x|f) \leq \int \left[\frac{1}{|\tilde{P}|} \sum_{i=1}^{|\tilde{P}|} 1 \right] d\nu(\tilde{P}, h|f) = 1;$$

3. $\overline{P}(x|f)$ sums to one, since

$$\begin{aligned}
\sum_x \overline{P}(x \mid f) &= \sum_x \left[\int \frac{1}{|\tilde{P}|} \sum_{i=1}^{|\tilde{P}|} P_i(x) d\nu(\tilde{P}, h|f) \right] \\
&= \int \frac{1}{|\tilde{P}|} \sum_{i=1}^{|\tilde{P}|} \left[\sum_x P_i(x) \right] d\nu(\tilde{P}, h|f) \\
&= \int \frac{1}{|\tilde{P}|} \sum_{i=1}^{|\tilde{P}|} 1 d\nu(\tilde{P}, h|f) \\
&= \int \frac{|\tilde{P}|}{|\tilde{P}|} d\nu(\tilde{P}, h|f) \\
&= 1.
\end{aligned}$$

Finally,

$$\begin{aligned}
\overline{P}(X \in t|f) &= \sum_x \mathbb{1}_{x \in t} \overline{P}(x|f) \\
&= \sum_x \left[\mathbb{1}_{x \in t} \int \mathbb{E}_{P \sim \mathcal{U}(\tilde{P})}[P(x)] d\nu(\tilde{P}, h|f) \right] \\
&= \sum_x \left[\mathbb{1}_{x \in t} \int \left[\frac{1}{|\tilde{P}|} \sum_{i=1}^{|\tilde{P}|} P_i(x) \right] d\nu(\tilde{P}, h|f) \right] \\
&= \int \frac{1}{|\tilde{P}|} \sum_{i=1}^{|\tilde{P}|} \left[\sum_x \mathbb{1}_{x \in t} P_i(x) \right] d\nu(\tilde{P}, h|f) \\
&= \mathbb{E}_{\tilde{P}, H} \left[\frac{1}{|\tilde{P}|} \sum_{i=1}^{|\tilde{P}|} \mathbb{E}_{X \sim P_i}[\mathbb{1}_{X \in t}] \middle| f \right] \\
&= \mathbb{E}_{\tilde{P}, H} \left[\frac{1}{|\tilde{P}|} \sum_{i=1}^{|\tilde{P}|} P_i(X \in t) \middle| f \right] \\
&= q(t, f).
\end{aligned}$$

□

Lemma 4. *If $X \perp T|F$, then*

$$\Pr(X \in T; \mathcal{A}) = \mathbb{E}_{T,F}[q(T,F)].$$

Proof. $\Pr(X \in T; \mathcal{A})$ denotes the probability that random variable X will be in target T (marginalized over all values of F) when T is random and X is drawn from $\overline{P}(X|F)$. Then,

$$\begin{aligned}
\Pr(X \in T; \mathcal{A}) &= \mathbb{E}_{T,X}\left[\mathbb{1}_{X\in T}|\mathcal{A}\right] \\
&= \mathbb{E}_{T}\left[\mathbb{E}_X[\mathbb{1}_{X\in T}|T,\mathcal{A}]\right] \\
&= \mathbb{E}_{T}\left[\mathbb{E}_F[\mathbb{E}_X[\mathbb{1}_{X\in T}|T,F,\mathcal{A}]|T]\right] \\
&= \mathbb{E}_{T}\left[\mathbb{E}_F[\mathbb{E}_X[\mathbb{1}_{X\in T}|F,\mathcal{A}]|T]\right] \\
&= \mathbb{E}_{T,F}\left[\mathbb{E}_X[\mathbb{1}_{X\in T}|F,\mathcal{A}]\right] \\
&= \mathbb{E}_{T,F}\left[\,\overline{P}(X \in T|F)\right] \\
&= \mathbb{E}_{T,F}\left[q(T,F)\right]
\end{aligned}$$

where the third equality makes use of the law of iterated expectation, the fourth follows from the conditional independence assumption, and the final equality follows from Lemma 3. □

Lemma 5. *(Maximum Number of Satisfying Vectors) Given an integer $1 \le k \le n$, a set $\mathcal{S} = \{\mathbf{s} : \mathbf{s} \in \{0,1\}^n, \|\mathbf{s}\| = \sqrt{k}\}$ of all n-length k-hot binary vectors, a set $\mathcal{P} = \{P : P \in \mathbb{R}^n, \sum_j P_j = 1\}$ of discrete n-dimensional simplex vectors, and a fixed scalar threshold $\epsilon \in [0,1]$, then for any fixed $P \in \mathcal{P}$,*

$$\sum_{\mathbf{s}\in\mathcal{S}} \mathbb{1}_{\mathbf{s}^\top P \ge \epsilon} \le \frac{1}{\epsilon}\binom{n-1}{k-1}$$

where $\mathbf{s}^\top P$ denotes the vector dot product between $\mathbf{s}$ and P.

Proof. For $\epsilon = 0$, the bound holds trivially. For $\epsilon > 0$, let S be a random quantity that takes values $\mathbf{s}$ uniformly in the set $\mathcal{S}$. Then, for any fixed $P \in \mathcal{P}$,

$$\begin{aligned}
\sum_{\mathbf{s}\in\mathcal{S}} \mathbb{1}_{\mathbf{s}^\top P \ge \epsilon} &= \binom{n}{k}\mathbb{E}\left[\mathbb{1}_{S^\top P \ge \epsilon}\right] \\
&= \binom{n}{k}\Pr\left(S^\top P \ge \epsilon\right).
\end{aligned}$$

Let $\mathbf{1}$ denotes the all ones vector. Under a uniform distribution on random quantity S and because

P does not change with respect to s, we have

$$\begin{aligned}
\mathbb{E}\left[S^\top P\right] &= \binom{n}{k}^{-1} \sum_{\mathbf{s}\in\mathcal{S}} \mathbf{s}^\top P \\
&= P^\top \binom{n}{k}^{-1} \sum_{\mathbf{s}\in\mathcal{S}} \mathbf{s} \\
&= P^\top \frac{\mathbf{1}\binom{n-1}{k-1}}{\binom{n}{k}} \\
&= P^\top \frac{\mathbf{1}\binom{n-1}{k-1}}{\frac{n}{k}\binom{n-1}{k-1}} \\
&= \frac{k}{n} P^\top \mathbf{1} \\
&= \frac{k}{n}
\end{aligned}$$

since P must sum to 1.

Noting that $S^\top P \geq 0$, we use Markov's inequality to get

$$\begin{aligned}
\sum_{\mathbf{s}\in\mathcal{S}} \mathbb{1}_{\mathbf{s}^\top P \geq \epsilon} &= \binom{n}{k} \Pr\left(S^\top P \geq \epsilon\right) \\
&\leq \binom{n}{k} \frac{1}{\epsilon} \mathbb{E}\left[S^\top P\right] \\
&= \binom{n}{k} \frac{1}{\epsilon}\frac{k}{n} \\
&= \frac{1}{\epsilon}\frac{k}{n}\frac{n}{k}\binom{n-1}{k-1} \\
&= \frac{1}{\epsilon}\binom{n-1}{k-1}.
\end{aligned}$$

□

Lemma 6. *(Maximum Proportion of Satisfying Strategies) Given an integer* $1 \leq k \leq n$*, a set* $\mathcal{S} = \{\mathbf{s} : \mathbf{s} \in \{0,1\}^n, \|\mathbf{s}\| = \sqrt{k}\}$ *of all* n*-length* k*-hot binary vectors, a set* $\mathcal{P} = \{P : P \in \mathbb{R}^n, \sum_j P_j = 1\}$ *of discrete* n*-dimensional simplex vectors, and a fixed scalar threshold* $\epsilon \in [0,1]$*, then*

$$\max_{\mathbf{s}\in\mathcal{S}} \frac{\mu(\mathcal{G}_{\mathbf{s},\epsilon})}{\mu(\mathcal{P})} \leq \frac{1}{\epsilon}\frac{k}{n}$$

where $\mathcal{G}_{\mathbf{s},\epsilon} = \{P : P \in \mathcal{P}, \mathbf{s}^\top P \geq \epsilon\}$ *and* μ *is Lebesgue measure.*

Proof. Similar results have been proved by others with regard to No Free Lunch theorems [17, 18, 21, 71, 95]. Our result concerns the maximum proportion of sufficiently good strategies (not the mean performance of strategies over all problems, as in the NFL case) and is a simplification over previous search-for-a-search results.

For $\epsilon = 0$, the bound holds trivially. For $\epsilon > 0$, We first notice that the $\mu(\mathcal{P})^{-1}$ term can be viewed as a uniform density over the region of the simplex $\mathcal{P}$, so that the integral becomes an expectation with respect to this distribution, where P is drawn uniformly from $\mathcal{P}$. Thus, for any $\mathbf{s} \in \mathcal{S}$,

$$\begin{aligned}\frac{\mu(\mathcal{G}_{\mathbf{s},\epsilon})}{\mu(\mathcal{P})} &= \int_{\mathcal{P}} \frac{1}{\mu(\mathcal{P})} \left[\mathbb{1}_{\mathbf{s}^\top P \geq \epsilon}\right] d\mu(P) \\ &= \mathbb{E}_{P\sim\mathcal{U}(\mathcal{P})}\left[\mathbb{1}_{\mathbf{s}^\top P \geq \epsilon}\right] \\ &= \Pr(\mathbf{s}^\top P \geq \epsilon) \\ &\leq \frac{1}{\epsilon}\mathbb{E}_{P\sim\mathcal{U}(\mathcal{P})}\left[\mathbf{s}^\top P\right],\end{aligned}$$

where the final line follows from Markov's inequality. Since the symmetric Dirichlet distribution in n dimensions with parameter $\alpha = 1$ gives the uniform distribution over the simplex, we get

$$\begin{aligned}\mathbb{E}_{P\sim\mathcal{U}(\mathcal{P})}\left[P\right] &= \mathbb{E}_{P\sim\text{Dir}(\alpha=1)}\left[P\right] \\ &= \left(\frac{\alpha}{\sum_{i=1}^{n}\alpha}\right)\mathbf{1} \\ &= \left(\frac{1}{n}\right)\mathbf{1},\end{aligned}$$

where $\mathbf{1}$ denotes the all ones vector. We have

$$\begin{aligned}\frac{1}{\epsilon}\mathbb{E}_{P\sim\mathcal{U}(\mathcal{P})}\left[\mathbf{s}^\top P\right] &= \frac{1}{\epsilon}\mathbf{s}^\top\mathbb{E}_{P\sim\mathcal{U}(\mathcal{P})}\left[P\right] \\ &= \frac{1}{\epsilon}\mathbf{s}^\top\left(\frac{1}{n}\right)\mathbf{1} \\ &= \frac{1}{\epsilon}\frac{k}{n}.\end{aligned}$$

□

5.2 The Fraction of Favorable Targets

Under our search framework and given an algorithm $\mathcal{A}$ and a fixed information object F, we can ask what fraction of possible target sets are *favorable* to $\mathcal{A}$, namely, for which $\mathcal{A}$ has a probability of success greater than random chance. Because the probability of chance success is proportional to the size of T, we define $p = |T|/|\Omega|$, and use *active information of expectations* ($I_{q(T,F)} := -\log p/q(T,F)$) to transform the ratio of (expected) success probabilities into bits [17, 57], where p is the per-query probability of success for uniform random sampling with replacement, and $q(T,F)$ is the expected per-query probability of success for algorithm $\mathcal{A}$ on the same problem[2]. For algorithm $\mathcal{A}$, positive $I_{q(T,F)}$ is equivalent to sampling a smaller search space for a target set of the same size in expectation, thus having a natural advantage over uniform search on the original search space. We can quantify favorability in terms of the number of bits advantage $\mathcal{A}$ has over uniform sampling. We have the following theorem concerning the proportion of b-bit favorable problems:

Theorem 1. *Let* $\tau = \{T \mid T \subseteq \Omega\}$ *and* $\tau_b = \{T \mid \emptyset \neq T \subseteq \Omega, I_{q(T,F)} \geq b\}$*. Then for* $b \geq 3$*,*

$$\frac{|\tau_b|}{|\tau|} \leq 2^{-b}.$$

Proof. First, by the definition of active information of expectations, $I_{q(T,F)} \geq b$ implies $|T| \leq \frac{|\Omega|}{2^b}$, since $q(T,F) \leq 1$. Thus,

$$\tau_b \subseteq \tau'_b = \left\{T \mid T \subseteq \Omega, 1 \leq |T| \leq \frac{|\Omega|}{2^b}\right\}. \tag{5.11}$$

For $|\Omega| < 2^b$ and $I_{q(T,F)} \geq b$, we have $|T| < 1$ for all elements of τ'_b (making the set empty) and the theorem follows immediately. Thus, $|\Omega| \geq 2^b$ for the remainder.

By Lemma 2, we have

$$\begin{align}
\frac{|\tau_b|}{|\tau|} &\leq \frac{|\tau'_b|}{|\tau|} \tag{5.12}\\
&= 2^{-|\Omega|} \sum_{k=0}^{\left\lfloor \frac{|\Omega|}{2^b} \right\rfloor} \binom{|\Omega|}{k} \tag{5.13}\\
&\leq 2^{-|\Omega|} 2^{|\Omega|-b} \tag{5.14}\\
&= 2^{-b}. \tag{5.15}
\end{align}$$

□

Thus, for a fixed information resource F, few target sets can be greatly favorable for any single algorithm.

Since this results holds for all target sets on Ω, a question immediately arises: would this scarcity of favorable problems exist if we limited ourselves to only sparse target sets, namely those having few target elements? Since the problems would be more difficult for uniform sampling, perhaps the fraction of favorable problems can grow accordingly. We examine this situation next.

[2]This pointwise KL-divergence is almost identical to the *active information* (I_+) transform defined in [17], but differs in that for active information q is the probability of success for algorithm $\mathcal{A}$ under some query constraint. Here we keep the pointwise KL-divergence form of I_+, but take the ratio of the expected per-query probabilities of success.

5.3 The Famine of Forte

If we restrict our consideration to k-sparse target sets, we have the following result which shows that a similarly restrictive bound continues to hold in the k-sparse case.

Theorem 2. *(Famine of Forte) Define*

$$\tau_k = \{T \mid T \subseteq \Omega, |T| = k \in \mathbb{N}\}$$

and let $\mathcal{B}_m$ denote any set of binary strings, such that the strings are of length m or less. Let

$$R = \{(T, F) \mid T \in \tau_k, F \in \mathcal{B}_m\}, \text{ and}$$
$$R_{q_{min}} = \{(T, F) \mid T \in \tau_k, F \in \mathcal{B}_m, q(T, F) \geq q_{min}\},$$

where $q(T, F)$ is the expected per-query probability of success for algorithm $\mathcal{A}$ on problem $\langle \Omega, T, F \rangle$. Then for any $m \in \mathbb{N}$,

$$\frac{|R_{q_{min}}|}{|R|} \leq \frac{p}{q_{min}}$$

and

$$\lim_{m \to \infty} \frac{|R_{q_{min}}|}{|R|} \leq \frac{p}{q_{min}}$$

where $p = k/|\Omega|$.

Thus, the famine of favorable problems exists for target-sparse problems as well. This result improves on Theorem 1 by removing the restrictions on the minimum search space size and amount of favorability, and by allowing for consideration of not just a fixed information resource F but any finite set of external information resources. We will now prove the result.

Proof. We begin by defining a set $\mathcal{S}$ of all $|\Omega|$-length target functions with exactly k ones, namely, $\mathcal{S} = \{\mathbf{s} : \mathbf{s} \in \{0,1\}^{|\Omega|}, \|\mathbf{s}\| = \sqrt{k}\}$. For each of these, we have $|\mathcal{B}_m|$ external information resources. The total number of search problems is therefore

$$\binom{|\Omega|}{k} |\mathcal{B}_m|. \tag{5.16}$$

We seek to bound the proportion of possible search problems for which $q(\mathbf{s}, f) \geq q_{\min}$ for any threshold $q_{\min} \in (0, 1]$. Thus,

$$\frac{|R_{q_{\min}}|}{|R|} \leq \frac{|\mathcal{B}_m| \sup_f \left[\sum_{\mathbf{s} \in \mathcal{S}} \mathbb{1}_{q(\mathbf{s},f) \geq q_{\min}}\right]}{|\mathcal{B}_m| \binom{|\Omega|}{k}} \tag{5.17}$$

$$= \binom{|\Omega|}{k}^{-1} \sum_{\mathbf{s} \in \mathcal{S}} \mathbb{1}_{q(\mathbf{s},f^*) \geq q_{\min}}, \tag{5.18}$$

where $f^* \in \mathcal{B}_m$ denotes the arg sup of the expression. Therefore,

$$\begin{aligned}\frac{|R_{q_{\min}}|}{|R|} &\leq \binom{|\Omega|}{k}^{-1} \sum_{\mathbf{s}\in\mathcal{S}} \mathbb{1}_{q(\mathbf{s},f^*)\geq q_{\min}} \\ &= \binom{|\Omega|}{k}^{-1} \sum_{\mathbf{s}\in\mathcal{S}} \mathbb{1}_{\overline{P}(\omega\in\mathbf{s}|f^*)\geq q_{\min}} \\ &= \binom{|\Omega|}{k}^{-1} \sum_{\mathbf{s}\in\mathcal{S}} \mathbb{1}_{\mathbf{s}^\top \overline{P}_{f^*}\geq q_{\min}}\end{aligned}$$

where the first equality follows from Lemma 3, $\omega \in$ s means the target function s evaluated at ω is one, and $\overline{P}_{f^*}$ represents the $|\Omega|$-length probability vector defined by $\overline{P}(\cdot|f^*)$. By Lemma 5, we have

$$\begin{aligned}\binom{|\Omega|}{k}^{-1} \sum_{\mathbf{s}\in\mathcal{S}} \mathbb{1}_{\mathbf{s}^\top \overline{P}_{f^*}\geq q_{\min}} &\leq \binom{|\Omega|}{k}^{-1} \left[\frac{1}{q_{\min}}\binom{|\Omega|-1}{k-1}\right] \\ &= \frac{k}{|\Omega|}\frac{1}{q_{\min}} \\ &= p/q_{\min} \end{aligned} \tag{5.19}$$

proving the result for finite external information resources.

To extend to infinite external information resources, let $A_m = \{f : f \in \{0,1\}^\ell, \ell \in \mathbb{N}, \ell \leq m\}$ and define

$$a_m := \frac{|A_m| \sup_{f\in A_m} \left[\sum_{\mathbf{s}\in\mathcal{S}} \mathbb{1}_{q(\mathbf{s},f)\geq q_{\min}}\right]}{|A_m|\binom{|\Omega|}{k}}, \tag{5.20}$$

$$b_m := \frac{|\mathcal{B}_m| \sup_{f\in \mathcal{B}_m} \left[\sum_{\mathbf{s}\in\mathcal{S}} \mathbb{1}_{q(\mathbf{s},f)\geq q_{\min}}\right]}{|\mathcal{B}_m|\binom{|\Omega|}{k}}. \tag{5.21}$$

We have shown that $a_m \leq p/q_{\min}$ for each $m \in \mathbb{N}$. Thus,

$$\begin{aligned}\limsup_{m\to\infty} \frac{|A_m| \sup_{f\in A_m} \left[\sum_{\mathbf{s}\in\mathcal{S}} \mathbb{1}_{q(\mathbf{s},f)\geq q_{\min}}\right]}{|A_m|\binom{|\Omega|}{k}} &= \limsup_{m\to\infty} a_m \\ &\leq \sup_m a_m \\ &\leq p/q_{\min}.\end{aligned}$$

Next, we use the monotone convergence theorem to show the limit exists. First,

$$\lim_{m\to\infty} a_m = \lim_{m\to\infty} \frac{\sup_{f\in A_m} \left[\sum_{\mathbf{s}} \mathbb{1}_{q(\mathbf{s},f)\geq q_{\min}}\right]}{\binom{|\Omega|}{k}} \tag{5.22}$$

By construction, the successive A_m are nested with increasing m, so the sequence of suprema (and numerator) are increasing, though not necessarily strictly increasing. The denominator is not dependent on m, so $\{a_m\}$ is an increasing sequence. Because it is also bounded above by p/q_{min}, the limit exists by monotone convergence. Thus,

$$\lim_{m\to\infty} a_m = \limsup_{m\to\infty} a_m \leq p/q_{\text{min}}.$$

Lastly,

$$\begin{aligned}\lim_{m\to\infty} b_m &= \lim_{m\to\infty} \frac{|\mathcal{B}_m| \sup_{f\in\mathcal{B}_m}\left[\sum_{\mathbf{s}\in\mathcal{S}} \mathbb{1}_{q(\mathbf{s},f)\geq q_{\text{min}}}\right]}{|\mathcal{B}_m|\binom{|\Omega|}{k}} \\ &= \lim_{m\to\infty} \frac{\sup_{f\in\mathcal{B}_m}\left[\sum_{\mathbf{s}\in\mathcal{S}} \mathbb{1}_{q(\mathbf{s},f)\geq q_{\text{min}}}\right]}{\binom{|\Omega|}{k}} \\ &\leq \lim_{m\to\infty} \frac{\sup_{f\in A_m}\left[\sum_{\mathbf{s}\in\mathcal{S}} \mathbb{1}_{q(\mathbf{s},f)\geq q_{\text{min}}}\right]}{\binom{|\Omega|}{k}} \\ &= \lim_{m\to\infty} a_m \\ &\leq p/q_{\text{min}}.\end{aligned}$$

□

We see that for small p (problems with sparse target sets) favorable search problems are rare if we desire a strong probability of success. The larger q_{min}, the smaller the proportion. In many real-world settings, we are given a difficult search problem (with minuscule p) and we hope that our algorithm has a reasonable chance of achieving success within a limited number of queries. According to this result, the proportion of problems fulfilling such criteria is also minuscule. Only if we greatly relax the minimum performance demanded, so that q_{min} approaches the scale of p, do such accommodating search problems become plentiful.

5.3.1 Corollary

Using the active information of expectations transform, we can restate the result in Theorem 2 to compare it directly with the bound from Theorem 1. Doing so shows the exact same bound continues to hold:

Corollary 1. *Let $R = \{(T, F) \mid T \in \tau_k, F \in \mathcal{B}_m\}$ and $R_b = \{(T, F) \mid T \in \tau_k, F \in \mathcal{B}_m, I_{q(T,F)} \geq b\}$. Then for any $m \in \mathbb{N}$*

$$\frac{|R_b|}{|R|} \leq 2^{-b}.$$

Proof. The proof follows from the definition of active information of expectations and Theorem 2. Note,

$$b \leq -\log_2\left(\frac{p}{q(T,F)}\right) \tag{5.23}$$

implies

$$q(T, F) \geq p2^b. \tag{5.24}$$

Since $I_{q(T,F)} \geq b$ implies $q(T, F) \geq p2^b$, the set of problems for which $I_{q(T,F)} \geq b$ can be no bigger than the set for which $q(T, F) \geq p2^b$. By Theorem 2, the proportion of problems for which $q(T, F)$ is at least $p2^b$ is no greater than $p/(p2^b)$. Thus,

$$\frac{|R_b|}{|R|} \leq \frac{1}{2^b}. \tag{5.25}$$

□

Thus, restricting ourselves to target-sparse search problems does nothing to increase the proportion of b-bit favorable problems, contrary to initial expectations. Furthermore, we see that finding a search problem for which an algorithm effectively reduces the search space by b bits requires at least b bits, so information is conserved in this context. Assuming you have no domain knowledge to guide the process of finding a search problem for which your search algorithm excels, you are unlikely to stumble upon one under uniform chance; indeed, they are exponentially rare in the amount of improvement sought.

5.3.2 Additional Corollary

If we define $\tilde{q}$ as the per-query probability of success (in contrast to the *expected* per-query probability of success $q(T, F)$), we can also bound the proportion of search problems with expected improvement over uniform random sampling, in the following way:

Corollary 2. *(Conservation of Expected $I_{\tilde{q}}$) Define*

$$I_{\tilde{q}} := -\log_2 p/\tilde{q},$$

where p is the per-query probability of success for uniform random sampling and $\tilde{q}$ is the per-query probability of success for an alternative search algorithm. Define

$$\tau_k = \{T \mid T \subseteq \Omega, |T| = k \in \mathbb{N}\}$$

and let $\mathcal{B}_m$ denote any set of binary strings, such that the strings are of length m or less. Let

$$R = \{(T, F) \mid T \in \tau_k, F \in \mathcal{B}_m\}, \text{ and}$$
$$R_b = \{(T, F) \mid T \in \tau_k, F \in \mathcal{B}_m, \mathbb{E}[I_{\tilde{q}}] \geq b\}.$$

Then for any $m \in \mathbb{N}$

$$\frac{|R_b|}{|R|} \leq 2^{-b}.$$

Proof. By Jensen's inequality and the concavity of $\log_2(\tilde{q}/p)$ in $\tilde{q}$, we have

$$\begin{aligned} b &\leq \mathbb{E}\left[-\log_2\left(\frac{p}{\tilde{q}}\right)\right] \\ &= \mathbb{E}\left[\log_2\left(\frac{\tilde{q}}{p}\right)\right] \\ &\leq \log_2\left(\frac{\mathbb{E}\left[\tilde{q}\right]}{p}\right) \\ &= -\log_2\left(\frac{p}{q(T,F)}\right) \\ &= I_{q(T,F)}. \end{aligned}$$

The result follows by invoking Corollary 1. □

5.4 Bound on Expected Efficiency

The results proven thus far can also be used in rederiving existing results from the literature. As one example, we will prove an early result by Culberson [15] using our theorems, which shows the proportion of problems for which a fixed algorithm is guaranteed to find a single target (in expectation) within ξ steps is bounded by $\xi/|\Omega|$. We prove this bound on the expected efficiency of algorithms as a corollary to the Famine of Forte.

Corollary 3. *(Expected Efficiency) Define*

$$\tau_1 = \{T \mid T \subseteq \Omega, |T| = 1\}$$

and let $\mathcal{B}_m$ denote any set of binary strings, such that the strings are of length m or less, for some $m \in \mathbb{N}$. Let R be the set of possible search problems on these sets and let $R_{\mathcal{A}}(\xi)$ be the subset of search problems for algorithm $\mathcal{A}$ such that the target element is found in expectation within ξ evaluations, namely,

$$\begin{aligned} R &= \{(T,F) \mid T \in \tau_1, F \in \mathcal{B}_m\}, \text{ and} \\ R_{\mathcal{A}}(\xi) &= \left\{(T,F) \mid T \in \tau_1, F \in \mathcal{B}_m, \mathbb{E}\left[\sum_{i=1}^{\xi} \mathbb{1}_{\omega_i \in T} \middle| F\right] \geq 1\right\}. \end{aligned}$$

Then,

$$\frac{|R_{\mathcal{A}}(\xi)|}{|R|} \leq \frac{\xi}{|\Omega|}.$$

Proof. We consider only algorithms that run for exactly ξ queries and terminate. For any algorithm terminating in fewer queries, we consider a related algorithm that has identical initial behavior but repeats the final query until ξ queries are achieved. Similarly, for algorithms that might produce more than ξ queries, we replace it with an algorithm that has identical initial behavior, but terminates after the ξth query.

Using the law of iterated expectation, we have

$$1 \leq \mathbb{E}\left[\sum_{i=1}^{\xi} \mathbb{1}_{\omega_i \in T} \middle| F\right] \tag{5.26}$$

$$= \mathbb{E}_{\tilde{P},H}\left[\mathbb{E}\left[\sum_{i=1}^{\xi} \mathbb{1}_{\omega_i \in T} \middle| \tilde{P}, H\right] \middle| F\right] \tag{5.27}$$

$$= \mathbb{E}_{\tilde{P},H}\left[\sum_{i=1}^{\xi} \mathbb{E}_{\omega_i}\left[\mathbb{1}_{\omega_i \in T} \middle| \tilde{P}, H\right] \middle| F\right] \tag{5.28}$$

$$= \mathbb{E}_{\tilde{P},H}\left[\sum_{i=1}^{\xi} P_i'(\omega \in T) \middle| F\right], \tag{5.29}$$

which, by linearity of expectation, implies

$$\mathbb{E}_{\tilde{P},H}\left[\frac{1}{\xi}\sum_{i=1}^{\xi} P_i'(\omega \in T) \middle| F\right] = q'(T, F) \tag{5.30}$$

$$\geq \frac{1}{\xi}. \tag{5.31}$$

Thus, $R_{\mathcal{A}}(\xi) = \{(T, F) \mid T \in \tau, F \in \mathcal{B}_m, q'(T, F) \geq \frac{1}{\xi}\}$ where $q'(T, F)$ is the expected per-query probability of success for some algorithm (possibly different from our original algorithm), $p = 1/|\Omega|$, and $q_{\min} = \frac{1}{\xi}$. Invoking Theorem 2 then gives the desired result, since it holds for all algorithms. □

5.5 The Famine of Favorable Strategies

The previous theorems have shown that strongly favorable search problems are rare for any fixed algorithm. However, not only are favorable problems rare, so are favorable strategies. Whether you hold fixed the algorithm and try to match a problem to it, or hold fixed the problem and try to match a strategy to it, both are provably difficult. Because matching problems to algorithms is hard, seemingly serendipitous agreement between the two calls for further explanation, serving as evidence against blind matching by independent mechanisms (especially for very sparse targets embedded in very large spaces). More importantly, this result places hard quantitative constraints (similar to minimax bounds in statistical learning theory) on information costs for automated machine learning, which attempts to match learning algorithms to learning problems [35, 80]. Our next result bounds the proportion of favorable strategies, exactly matching the bound given in Theorem 2.

Theorem 3. *(The Famine of Favorable Strategies) Let* $p = |t|/|\Omega|$*. Then for any fixed search problem* (Ω, t, f)*, set of probability mass functions* $\mathcal{P} = \{P : P \in \mathbb{R}^{|\Omega|}, \sum_j P_j = 1\}$*, and a fixed threshold* $q_{min} \in [p, 1]$*,*

$$\frac{\mu(\mathcal{G}_{t,q_{min}})}{\mu(\mathcal{P})} \leq \frac{p}{q_{min}},$$

where $\mathcal{G}_{t,q_{min}} = \{P : P \in \mathcal{P}, t^\top P \geq q_{min}\}$ and μ is Lebesgue measure. Furthermore, the proportion of possible search strategies giving at least b bits of active information of expectations is no greater than 2^{-b}.

Proof. Applying Lemma 6, with $\mathrm{s} = t$, $\epsilon = q_{\min}$, $k = |t|$, $n = |\Omega|$, and $p = |t|/|\Omega|$, yields the first result, while following the same steps as Corollary 1 gives the second (noting that by Lemma 3 each strategy is equivalent to a corresponding $q(t, f)$). □

5.6 Learning Under Dependence

When p is very small, the previous results prove the difficulty of blindly encountering a problem for which your algorithm greatly outperforms random chance; most problems are not favorable for a given algorithm. However, when problems *are* favorable, what is it that makes them favorable? Here we are introduced to the importance of **dependence** in search. We show that only for problems where the external information resource F provides exploitable information concerning the target set T can an algorithm have a large single-query probability of success. This is proven next.

Theorem 4. *Define $\tau_k = \{T \mid T \subseteq \Omega, |T| = k \in \mathbb{N}\}$ and let $\mathcal{B}_m$ denote any set of binary strings, such that the strings are of length m or less. Define q as the expected per-query probability of success under the joint distribution on $T \in \tau_k$ and $F \in \mathcal{B}_m$ for any fixed algorithm $\mathcal{A}$, so that $q := \mathbb{E}_{T,F}\left[q(T, F)\right]$, namely,*

$$q = \mathbb{E}_{T,F}\left[\,\overline{P}(\omega \in T|F)\right] = \Pr(\omega \in T; \mathcal{A}).$$

Then,

$$q \leq \frac{I(T; F) + D(P_T \| \mathcal{U}_T) + 1}{I_\Omega}$$

where $I_\Omega = -\log_2 k/|\Omega|$, $D(P_T\|\mathcal{U}_T)$ is the Kullback-Liebler divergence between the marginal distribution on T and the uniform distribution on T, and $I(T; F)$ is the mutual information. Alternatively, we can write

$$q \leq \frac{H(\mathcal{U}_T) - H(T \mid F) + 1}{I_\Omega}$$

where $H(\mathcal{U}_T) = \log_2 \binom{|\Omega|}{k}$.

This result shows how the bound on single-query probability of success improves proportionally with the amount of reliable information given concerning targets. We quantify the degree of reliability as the mutual information between target set T and external information resource F, under any fixed joint distribution. The larger the mutual information, the more information F can give us regarding the location of T, and thus, the greater the opportunity for success. We see that I_Ω can be interpreted as the information cost of locating a target element in the absence of side-information. $D(P_T\|\mathcal{U}_T)$ is naturally interpreted as the predictability of the target sets, since large values imply the probable occurrence of only a small number of possible target sets.

The mutual information $I(T;F)$ is the amount of exploitable information the external resource contains regarding T; lowering the mutual information lowers the maximum expected probability of success for the algorithm. Lastly, the 1 in the numerator upper bounds the contribution of pure randomness. This expression constrains the relative contributions of predictability, problem difficulty, side-information, and randomness for a successful search.

Proof. This proof loosely follows that of Fano's Inequality [24], being a reversed generalization of it, so in keeping with the traditional notation we let $X := \omega$ for the remainder of this proof. Let $Z = \mathbb{1}(X \in T)$. Using the chain rule for entropy to expand $H(Z,T|X)$ in two different ways, we get

$$\begin{aligned} H(Z,T|X) &= H(Z|T,X) + H(T|X) && (5.32)\\ &= H(T|Z,X) + H(Z|X). && (5.33) \end{aligned}$$

By definition, $H(Z|T,X) = 0$, and by the data processing inequality $H(T|F) \leq H(T|X)$. Thus,

$$H(T|F) \leq H(T|Z,X) + H(Z|X). \tag{5.34}$$

Define $P_g = \Pr(X \in T; \mathcal{A}) = \Pr(Z = 1)$. Then,

$$\begin{aligned} H(T|Z,X) &= (1-P_g)H(T|Z=0,X) + P_g H(T|Z=1,X) && (5.35)\\ &\leq (1-P_g)\log_2\binom{|\Omega|}{k} + P_g \log_2\binom{|\Omega|-1}{k-1} && (5.36)\\ &= \log_2\binom{|\Omega|}{k} - P_g \log_2 \frac{|\Omega|}{k}. && (5.37) \end{aligned}$$

We let $H(\mathcal{U}_T) = \log_2\binom{|\Omega|}{k}$, being the entropy of the uniform distribution over k-sparse target sets in Ω. Therefore,

$$H(T|F) \leq H(\mathcal{U}_T) - P_g \log_2 \frac{|\Omega|}{k} + H(Z|X). \tag{5.38}$$

Using the definitions of conditional entropy and I_Ω, we get

$$H(T) - I(T;F) \leq H(\mathcal{U}_T) - P_g I_\Omega + H(Z|X), \tag{5.39}$$

which implies

$$\begin{aligned} P_g I_\Omega &\leq I(T;F) + H(\mathcal{U}_T) - H(T) + H(Z|X) && (5.40)\\ &= I(T;F) + D(P_T \| \mathcal{U}_T) + H(Z|X). && (5.41) \end{aligned}$$

Examining $H(Z|X)$, we see it captures how much entropy of Z is due to the randomness of T. To see this, imagine Ω is a roulette wheel and we place our bet on X. Target elements are "chosen" as balls land on random slots, according to the distribution on T. When a ball lands on X as often as not (roughly half the time), this quantity is maximized. Thus, this entropy captures the contribution of dumb luck, being averaged over all X. (When balls move towards always landing on X, something other than luck is at work.) We upperbound this by its maximum value of 1 and obtain

$$\Pr(X \in T; \mathcal{A}) \leq \frac{I(T;F) + D(P_T \| \mathcal{U}_T) + 1}{I_\Omega}, \tag{5.42}$$

and substitute q for $\Pr(X \in T; \mathcal{A})$ to obtain the first result, noting that $q = \mathbb{E}_{T,F}\left[\,\overline{P}(\omega \in T|F)\right]$ specifies a proper probability distribution by the linearity and boundedness of the expectation. To obtain the second form, use the definitions $I(T;F) = H(T) - H(T|F)$ and $D(P_T \| \mathcal{U}_T) = H(\mathcal{U}_T) - H(T)$. $\square$

5.7 Why Machine Learning Works in One Equation

Rather than simply upper-bounding the probability of success, we would like to know exactly how mutual information, target sparseness, target predictability and random luck affect the expected outcome. We thus prove the following theorem which gives a closed-form expression for the single-query probability of success given those factors:

Theorem 5. *(Probability of Success) Define*

$$\tau_k = \{T \mid T \subseteq \Omega, |T| = k \in \mathbb{N}\}$$

and let $\mathcal{B}_m$ denote any set of binary strings, such that the strings are of length m or less. Let $X \in \Omega$ be drawn according to algorithm $\mathcal{A}$ and define $Z = \mathbb{1}(X \in T)$. For any joint distribution on $T \in \tau_k$ and information resource $F \in \mathcal{B}_m$, if $H(T|Z=0,X) \neq H(T|Z=1,X)$, then

$$\Pr(X \in T; \mathcal{A}) = \frac{I(T;F) - I_L + D(P_T \| \mathcal{U}_T) + H(Z|X) + Cr}{I_\Omega + \mathbb{E}_X\left[D(P_{T|Z=1,X} \| \mathcal{U}_{T|Z=1,X})\right] + Cr} \tag{5.43}$$

where $I_\Omega = -\log_2 k/|\Omega|$, $I_L = I(T;F) - I(T;X)$ is the information leakage, $D(P_T \| \mathcal{U}_T)$ is the Kullback-Liebler divergence between the marginal distribution on T and the uniform distribution on T, $I(T;F)$ is the mutual information between T and F, and $Cr = \log_2(1 - k/|\Omega|) - \mathbb{E}_X\left[D(P_{T|Z=0,X} \| \mathcal{U}_{T|Z=0,X})\right]$.

Proof. Define $Z = \mathbb{1}(X \in T)$. Using the chain rule for entropy to expand $H(Z,T|X)$ in two different ways, we get

$$\begin{aligned} H(Z,T|X) &= H(Z|T,X) + H(T|X) && (5.44)\\ &= H(T|Z,X) + H(Z|X). && (5.45) \end{aligned}$$

By definition, $H(Z|T,X) = 0$, which gives

$$H(T|X) = H(T|Z,X) + H(Z|X). \tag{5.46}$$

Define $P_g = \Pr(X \in T; \mathcal{A}) = \Pr(Z = 1)$. Then,

$$\begin{aligned} H(T|Z,X) &= (1 - P_g)H(T|Z=0,X) + P_g H(T|Z=1,X) && (5.47)\\ &= H(T|Z=0,X) - P_g\left[H(T|Z=0,X) - H(T|Z=1,X)\right]. && (5.48) \end{aligned}$$

Combining this result with Equation 5.46, we get

$$H(T|X) = H(T|Z=0,X) - P_g\left[H(T|Z=0,X) - H(T|Z=1,X)\right] + H(Z|X). \tag{5.49}$$

By the definition of mutual information, $H(T|X) = H(T) - I(T;X)$, so we can substitute and rearrange to obtain

$$P_g = \frac{I(T;X) - H(T) + H(T|Z=0,X) + H(Z|X)}{H(T|Z=0,X) - H(T|Z=1,X)}. \tag{5.50}$$

Let $\mathcal{U}_T$ denote the uniform distribution over $T \in \tau_k$. Using the facts that

$$H(T) = \log_2 \binom{|\Omega|}{k} - D(P_T \| \mathcal{U}_T), \tag{5.51}$$

$$H(T|Z=0,X) = \sum_{x\in\Omega} \Pr(X=x) H(T|Z=0, X=x), \tag{5.52}$$

$$= \sum_{x\in\Omega} \Pr(X=x)\left[\log_2 \binom{|\Omega|-1}{k} - D(P_{T|Z=0,X=x} \| \mathcal{U}_{T|Z=0,X=x})\right], \tag{5.53}$$

$$= \log_2 \binom{|\Omega|-1}{k} - \mathbb{E}_X\left[D(P_{T|Z=0,X} \| \mathcal{U}_{T|Z=0,X})\right], \tag{5.54}$$

we get

$$P_g = \frac{I(T;X) - \log_2\binom{|\Omega|}{k} + D(P_T\|\mathcal{U}_T) + \log_2\binom{|\Omega|-1}{k} - \mathbb{E}_X\left[D(P_{T|Z=0,X}\|\mathcal{U}_{T|Z=0,X})\right] + H(Z|X)}{H(T|Z=0,X) - H(T|Z=1,X)} \tag{5.55}$$

$$= \frac{I(T;X) + \log_2\left(1 - \frac{k}{|\Omega|}\right) + D(P_T\|\mathcal{U}_T) - \mathbb{E}_X\left[D(P_{T|Z=0,X}\|\mathcal{U}_{T|Z=0,X})\right] + H(Z|X)}{H(T|Z=0,X) - H(T|Z=1,X)}. \tag{5.56}$$

Applying similar substitutions to the denominator, we obtain

$$P_g = \frac{I(T;X) + \log_2\left(1 - \frac{k}{|\Omega|}\right) + D(P_T\|\mathcal{U}_T) - \mathbb{E}_X\left[D(P_{T|Z=0,X}\|\mathcal{U}_{T|Z=0,X})\right] + H(Z|X)}{\log_2\binom{|\Omega|-1}{k} - \log_2\binom{|\Omega|-1}{k-1} + \mathbb{E}_X\left[D(P_{T|Z=1,X}\|\mathcal{U}_{T|Z=1,X})\right] - \mathbb{E}_X\left[D(P_{T|Z=0,X}\|\mathcal{U}_{T|Z=0,X})\right]} \tag{5.57}$$

$$= \frac{I(T;X) + \log_2\left(1 - \frac{k}{|\Omega|}\right) + D(P_T\|\mathcal{U}_T) - \mathbb{E}_X\left[D(P_{T|Z=0,X}\|\mathcal{U}_{T|Z=0,X})\right] + H(Z|X)}{\log_2\left(\frac{|\Omega|}{k} - 1\right) + \mathbb{E}_X\left[D(P_{T|Z=1,X}\|\mathcal{U}_{T|Z=1,X})\right] - \mathbb{E}_X\left[D(P_{T|Z=0,X}\|\mathcal{U}_{T|Z=0,X})\right]} \tag{5.58}$$

$$= \frac{I(T;X) + \log_2\left(1 - \frac{k}{|\Omega|}\right) + D(P_T\|\mathcal{U}_T) - \mathbb{E}_X\left[D(P_{T|Z=0,X}\|\mathcal{U}_{T|Z=0,X})\right] + H(Z|X)}{I_\Omega + \log_2\left(1 - \frac{k}{|\Omega|}\right) + \mathbb{E}_X\left[D(P_{T|Z=1,X}\|\mathcal{U}_{T|Z=1,X})\right] - \mathbb{E}_X\left[D(P_{T|Z=0,X}\|\mathcal{U}_{T|Z=0,X})\right]}. \tag{5.59}$$

Let $Cr = \log_2\left(1 - \frac{k}{|\Omega|}\right) - \mathbb{E}_X\left[D(P_{T|Z=0,X}\|\mathcal{U}_{T|Z=0,X})\right]$, as a correction term and define the *information leakage* $I_L = I(T;F) - I(T;X)$. Using the definition of P_g and adding/subtracting $I(T;F)$ from the numerator, we conclude

$$\Pr(X \in T; \mathcal{A}) = \frac{I(T;F) - I_L + D(P_T\|\mathcal{U}_T) + H(Z|X) + Cr}{I_\Omega + \mathbb{E}_X\left[D(P_{T|Z=1,X}\|\mathcal{U}_{T|Z=1,X})\right] + Cr}. \tag{5.60}$$

□

The elements of Equation 5.43 suggest the following interpretation:

- $I(T;F)$ — The mutual information between the target set and the external information resource;
- I_L — The amount of mutual information concerning the target set ignored or lost by the learning algorithm when constructing its distribution from the external information resource (e.g., building its model from the training data);
- $D(P_T\|\mathcal{U}_T)$ — The overall predictability of the target set (when this is large, T becomes more predictable and T becomes maximally unpredictable when this quantity is zero);
- $H(Z|X)$ — The contribution of pure random chance to the search process;
- I_Ω — The baseline problem difficulty, measured in bits;
- $\mathbb{E}_X\left[D(P_{T|Z=1,X}\|\mathcal{U}_{T|Z=1,X})\right]$ — Expected structural predictability term, signifying on average how much information you gain about T as a set given you know the location of one element in T; and
- Cr — The correction term, which balances the equation to make it a valid probability.

Equation 5.43 gives an answer to the question "why does machine learning work?," as a closed-form single equation. The equation contains many components that must be known or estimated, which could prove difficult in practice. However, given some assumptions of distributions and search process, the single-query probability of success can be computed exactly, as is done for examples given in Chapter 8.

To complete our set of theoretical results for our search framework, we will examine a seminal contribution to the view of machine learning as search, showing the need for biases in the search process. We formalize a result given by Mitchell [55], extending it to hold for multi-class classification problems (the original statement and informal proof were given with respect to binary classification).

5.8 The Need for Biased Classifiers

Mitchell demonstrated the need for binary classification algorithms to hold certain assumptions concerning the problems they are applied to [55]. Defining *inductive bias* as any preferences in an algorithm based on something other than strict consistency with training data, Mitchell demonstrated that an algorithm with no inductive bias necessarily has generalization performance equivalent to random guessing. He did this by arguing that if we retain a space of all hypotheses (binary labelings) that are consistent with training data, beginning with all possible hypotheses on the instance space, then those hypotheses that remain will have an equal number of positive labels and negative labels for any instance not yet seen. If strict consistency with training data is our only criterion for choosing hypotheses, we cannot decide if an unseen instance should receive a positive or negative label based on majority voting, since the remaining consistent hypotheses are equally split among both outcomes. Thus, we must essentially guess, having an equal chance of being correct or wrong in each instance. While Mitchell gave an informal argument similar to the preceding to prove this point, we formally prove the need for biased classifiers, extending the result to hold for all finite multi-class learning problems.

We will give two theorems, one for when the true concept h^* is fixed, and the second for the more general case, when the true concept is chosen according to some distribution. The first is a special case of the second (assuming a point-mass, degenerate distribution), but the proof for the special case (with credit to Cosma Shalizi) is much simpler, so we begin with it.

5.8.1 Special Case: Concept and Test Set Fixed

Theorem 6. *Define as follows:*

- $\mathcal{X}$ *- finite instance space,*
- $\mathcal{Y}$ *- finite label space,*
- Ω - $\mathcal{Y}^{\mathcal{X}}$*, the space of possible concepts on* $\mathcal{X}$*,*
- h *- a hypothesis,* $h \in \Omega$*,*
- h^* *- the true concept,* $h^* \in \Omega$*,*
- $D = \{(x_1, y_1), \ldots, (x_N, y_N)\}$ *- any training dataset where* $x_i \in \mathcal{X}$*,* $y_i \in \mathcal{Y}$*,*
- $D_x = \{x : (x, \cdot) \in D\}$ *- the set of* x *instances in* D*,*
- $V_x = \{x_1, \ldots, x_M\}$ *- any test dataset disjoint from* D_x *(i.e.,* $D_x \cap V_x = \emptyset$*) containing exactly* M *elements* $x_i \in \mathcal{X}$ *with* $M > 0$*,*
- Ω_D *- subset of hypotheses consistent with* D*, and*
- unbiased classifier $\mathcal{A}$ *- any classifier such that* $P(h|D, V_x) = \mathbb{1}(h \in \Omega_D)/|\Omega_D|$ *(i.e., makes no assumptions beyond strict consistency with training data).*

For fixed h^* *and* $V_x = v_x$*, the distribution of 0-1 generalization error counts for any unbiased classifier on true concept* h^* *is given by*

$$P(w|h^*, D, v_x; \mathcal{A}) = \binom{M}{w} \left(1 - \frac{1}{|\mathcal{Y}|}\right)^{w} \left(\frac{1}{|\mathcal{Y}|}\right)^{M-w}$$

where w *is the number of wrong predictions on test set* v_x*.*

Proof. We assume that $h^* \to d \to h$, implying $P(h|h^*, d, v_x) = P(h|d, v_x)$ by d-separation. Given h^*, the true label of every instance is $\mathcal{X}$ becomes fixed. Define,

$$w_{h,h^*}(v_x) = \sum_{x \in v_x} \mathbb{1}(h(x) \neq h^*(x))), \tag{5.61}$$

$$R = \{h : h \in \Omega_D, \}, \text{ and} \tag{5.62}$$

$$R_w = \{h : h \in \Omega_D, w_{h,h^*}(v_x) = w\}. \tag{5.63}$$

Given these definitions, $|R| = |\mathcal{Y}|^{|\mathcal{X}|-|D|}$ and $|R_w| = \binom{|v_x|}{w}(|\mathcal{Y}|-1)^w|\mathcal{Y}|^{|\mathcal{X}|-|D|-|v_x|}$. Because unbiased classifier $\mathcal{A}$ has a uniform distribution on the set of all consistent hypotheses, the ratio $|R_w|/|R|$ is equivalent to the probability of choosing a hypothesis making exactly w errors on v_x, namely

$$\frac{|R_w|}{|R|} = \frac{\binom{|v_x|}{w}(|\mathcal{Y}|-1)^w|\mathcal{Y}|^{|\mathcal{X}|-|D|-|v_x|}}{|\mathcal{Y}|^{|\mathcal{X}|-|D|}} \tag{5.64}$$

$$= \binom{|v_x|}{w}\left[|\mathcal{Y}|\left(1-\frac{1}{|\mathcal{Y}|}\right)\right]^w \frac{1}{|\mathcal{Y}|^{|v_x|}} \tag{5.65}$$

$$= \binom{|v_x|}{w}\left(1-\frac{1}{|\mathcal{Y}|}\right)^w \frac{|\mathcal{Y}|^w}{|\mathcal{Y}|^{|v_x|}} \tag{5.66}$$

$$= \binom{|v_x|}{w}\left(1-\frac{1}{|\mathcal{Y}|}\right)^w \left(\frac{1}{|\mathcal{Y}|}\right)^{|v_x|-w} \tag{5.67}$$

$$= \binom{M}{w}\left(1-\frac{1}{|\mathcal{Y}|}\right)^w \left(\frac{1}{|\mathcal{Y}|}\right)^{M-w}. \tag{5.68}$$

□

5.8.2 General Case: Concept and Test Set Drawn from Distributions

We again consider algorithms that make no assumptions beyond strict consistency with training data. This requires that the training data be noiseless, with accurate class labels. Although one can extend the proof to handle the case of noisy training labels (see Appendix A), doing so increases the complexity of the proof and does not add anything substantial to the discussion, since the same result continues to hold. Thus, we focus on the case of noiseless training data, assuming strict consistency with it.

Theorem 7. *Define as follows:*

- $\mathcal{X}$ *- finite instance space,*
- $\mathcal{Y}$ *- finite label space,*
- Ω *-* $\mathcal{Y}^{\mathcal{X}}$*, the space of possible concepts on* $\mathcal{X}$*,*
- h *- a hypothesis,* $h \in \Omega$*,*
- $D = \{(x_1, y_1), \ldots, (x_N, y_N)\}$ *- any training dataset where* $x_i \in \mathcal{X}$*,* $y_i \in \mathcal{Y}$*,*
- $D_x = \{x : (x, \cdot) \in D\}$ *- the set of* x *instances in* D*,*
- $V_x = \{x_1, \ldots, x_M\}$ *- any test dataset disjoint from* D_x *(i.e.,* $D_x \cap V_x = \emptyset$*) containing exactly* M *elements* $x_i \in \mathcal{X}$ *with* $M > 0$*, hidden from the algorithm during learning,*
- Ω_D *- subset of hypotheses strictly consistent with* D*, and*
- unbiased classifier $\mathcal{A}$ *- any classifier such that* $P(h|D, M) = \mathbb{1}(h \in \Omega_D)/|\Omega_D|$ *(i.e., makes no assumptions beyond strict consistency with training data).*

Then the distribution of 0-1 generalization error counts for any unbiased classifier is given by

$$P(w|D, M; \mathcal{A}) = \binom{M}{w}\left(1-\frac{1}{|\mathcal{Y}|}\right)^w \left(\frac{1}{|\mathcal{Y}|}\right)^{M-w}$$

where w is the number of wrong predictions on disjoint test sets of size M. Thus, every unbiased classifier has generalization performance equivalent to random guessing (e.g., flipping a coin) of class labels for unseen instances.

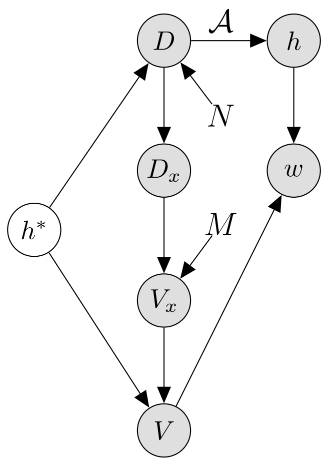


Figure 5.1: Graphical model of objects involved in unbiased classification and their relationships.

Proof. Figure 5.1 gives the graphical model for our problem setting. In agreement with this model, we assume the training data D are generated from the true concept h^* by some process, and that h is chosen by $\mathcal{A}$ using D alone, without access to h^*. Thus, $h^* \to D \to h$, implying $P(h^*|D, h, M) = P(h^*|D, M)$ by d-separation. By similar reasoning, since $\{D_x, M\} \to V_x$, with D as an ancestor of both V_x and h and no other active path between them, we have $P(V_x|f, D, h, M) = P(V_x|f, D, M)$. This can be verified intuitively by the fact that V_x is generated prior to the algorithm choosing h, and the algorithm has no access to V_x when choosing h (it only has access to D, which we're already conditioning on).

Let $K = \mathbb{1}(h \in \Omega_D)/|\Omega_D|$ and let L be the number of free instances in $\mathcal{X}$, namely

$$\begin{aligned} L &= |\mathcal{X}| - (|D_x| + |V_x|) && (5.69)\\ &= |\mathcal{X}| - (N + M). && (5.70) \end{aligned}$$

Then

$$P(w|D,M;\mathcal{A}) = \frac{P(w,D|M)}{P(D|M)} \tag{5.71}$$

$$= \frac{\sum_{h\in\Omega_D} P(w,h,D|M)}{\sum_{h\in\Omega_D} P(h,D|M)} \tag{5.72}$$

$$= \frac{\sum_{h\in\Omega_D} P(w|h,D,M)P(h|D,M)P(D|M)}{\sum_{h\in\Omega_D} P(h|D,M)P(D|M)} \tag{5.73}$$

$$= \frac{\sum_{h\in\Omega_D} P(w|h,D,M)KP(D|M)}{\sum_{h\in\Omega_D} KP(D|M)} \tag{5.74}$$

$$= \frac{KP(D|M)\sum_{h\in\Omega_D} P(w|h,D,M)}{KP(D|M)|\Omega_D|} \tag{5.75}$$

$$= \frac{1}{|\Omega_D|}\sum_{h\in\Omega_D} P(w|h,D,M) \tag{5.76}$$

$$= \frac{1}{|\mathcal{Y}|^{M+L}}\sum_{h\in\Omega_D} P(w|h,D,M) \tag{5.77}$$

$$= \frac{1}{|\mathcal{Y}|^{M+L}}\sum_{h\in\Omega_D} P(w|h,D,M), \tag{5.78}$$

Marginalizing over possible true concepts f for term $P(w|h,D,M)$ and letting $Z = \{f,h,D,M\}$, we have

$$P(w|h,D,M) = \sum_{f\in\Omega} P(w,f|h,D,M) \tag{5.79}$$

$$= \sum_{f\in\Omega} P(w|Z)P(f|h,D,M) \tag{5.80}$$

$$= \sum_{f\in\Omega} P(w|Z)P(f|D,M) \quad \text{(by d-separation)} \tag{5.81}$$

$$= \sum_{f\in\Omega} P(f|D,M)\sum_{v_x} P(w,v_x|Z) \tag{5.82}$$

$$= \sum_{f\in\Omega} P(f|D,M)\sum_{v_x} P(v_x|Z)P(w|Z,v_x) \tag{5.83}$$

$$= \sum_{f\in\Omega} P(f|D,M)\sum_{v_x} P(v_x|Z)\mathbb{1}(w = w_{h,f}(v_x)), \tag{5.84}$$

where $w_{h,f}(v_x) = \sum_{x\in v_x}\mathbb{1}(h(x) \neq f(x))$ and the final equality follows since

$$P(w|Z,v_x) = P(w|f,h,D,M,v_x) = \begin{cases} 1 & w = w_{h,f}(v_x), \\ 0 & w \neq w_{h,f}(v_x). \end{cases} \tag{5.85}$$

Combining (5.78) and (5.84), we obtain

$$P(w|D,M;\mathcal{A}) = \frac{1}{|\mathcal{Y}|^{M+L}} \sum_{h\in\Omega_D} \left[\sum_{f\in\Omega} P(f|D,M) \sum_{v_x} P(v_x|f,h,D,M) \mathbb{1}(w = w_{h,f}(v_x)) \right] \tag{5.86}$$

$$= \frac{1}{|\mathcal{Y}|^{M+L}} \sum_{h\in\Omega_D} \left[\sum_{f\in\Omega} P(f|D,M) \sum_{v_x} P(v_x|f,D,M) \mathbb{1}(w = w_{h,f}(v_x)) \right] \tag{5.87}$$

$$= \frac{1}{|\mathcal{Y}|^{M+L}} \sum_{f\in\Omega} P(f|D,M) \sum_{v_x} P(v_x|Z') \sum_{h\in\Omega_D} \mathbb{1}(w = w_{h,f}(v_x)), \tag{5.88}$$

where we have defined $Z' := \{f, D, M\}$ and the second equality follows from d-separation between V_x and h, conditioned on D.

Note that $\sum_{h\in\Omega_D} \mathbb{1}(w = w_{h,f}(v_x))$ is the number of hypotheses strictly consistent with D that disagree with concept f exactly w times on v_x. There are $\binom{M}{w}$ ways to choose w disagreements with f on v_x, and for each of the w disagreements we can choose $|\mathcal{Y}| - 1$ possible values for h at that instance, giving a multiplicative factor of $(|\mathcal{Y}| - 1)^w$. For the remaining L instances that are in neither D nor v_x, we have $|\mathcal{Y}|$ possible values, giving the additional multiplicative factor of $|\mathcal{Y}|^L$. Thus,

$$P(w|D,M;\mathcal{A}) = \frac{1}{|\mathcal{Y}|^{M+L}} \sum_{f\in\Omega} P(f|D,M) \sum_{v_x} P(v_x|Z') \left[\binom{M}{w} (|\mathcal{Y}| - 1)^w |\mathcal{Y}|^L \right] \tag{5.89}$$

$$= \frac{1}{|\mathcal{Y}|^{M+L}} \left[\binom{M}{w} (|\mathcal{Y}| - 1)^w |\mathcal{Y}|^L \right] \sum_{f\in\Omega} P(f|D,M) \sum_{v_x} P(v_x|Z') \tag{5.90}$$

$$= \frac{1}{|\mathcal{Y}|^M |\mathcal{Y}|^L} \left[\binom{M}{w} (|\mathcal{Y}| - 1)^w |\mathcal{Y}|^L \right] \sum_{f\in\Omega} P(f|D,M) \tag{5.91}$$

$$= \binom{M}{w} (|\mathcal{Y}| - 1)^w \frac{1}{|\mathcal{Y}|^M} \tag{5.92}$$

$$= \binom{M}{w} \left[|\mathcal{Y}| \left(1 - \frac{1}{|\mathcal{Y}|}\right) \right]^w \frac{1}{|\mathcal{Y}|^M} \tag{5.93}$$

$$= \binom{M}{w} \left(1 - \frac{1}{|\mathcal{Y}|}\right)^w \frac{|\mathcal{Y}|^w}{|\mathcal{Y}|^M} \tag{5.94}$$

$$= \binom{M}{w} \left(1 - \frac{1}{|\mathcal{Y}|}\right)^w |\mathcal{Y}|^{w-M} \tag{5.95}$$

$$= \binom{M}{w} \left(1 - \frac{1}{|\mathcal{Y}|}\right)^w \left(\frac{1}{|\mathcal{Y}|}\right)^{M-w}. \tag{5.96}$$

□

5.8.3 Discussion

As a historical note, Wolpert explored many related issues in [91], proving a related result using his Extended Bayesian Formalism. Namely, he showed that in the noiseless case for error $E = w/(|\mathcal{X}| - |D|)$,

$$P(E|h, D) = \binom{|\mathcal{X}| - |D|}{(|\mathcal{X}| - |D|)(1 - E)} (|\mathcal{Y}| - 1)^{E(|\mathcal{X}|-|D|)} / |\mathcal{Y}|^{(|\mathcal{X}|-|D|)}, \tag{5.97}$$

whenever $P(f|D)$ is uniform (i.e., in a maximum entropy universe), and argues by symmetry that a similar result would hold for $P(h|D)$ being uniform. By letting $M = |\mathcal{X}| - |D|$ and taking into account that $E = w/(|\mathcal{X}| - |D|)$, one can recover a result matching (5.96) from (5.97), namely

$$P(w|h, D, M) = P(E|h, D) \tag{5.98}$$

$$= \binom{M}{w} \left(1 - \frac{1}{|\mathcal{Y}|}\right)^{w} \left(\frac{1}{|\mathcal{Y}|}\right)^{M-w}. \tag{5.99}$$

While we consider different quantities here (i.e., $P(w|D, M; \mathcal{A})$) and prove our results using uniformity in the algorithm rather than in the universe (i.e., $P(h|D, M)$ being uniform rather than $P(f|D)$), the similar proof techniques and closely related results demonstrate the continued relevance of that early work.

These results show that if the learning algorithm makes no assumptions beyond strict consistency with the training data (i.e., equally weighting every consistent hypothesis) then the generalization ability of the algorithm for previously unseen training data cannot surpass that of random uniform guessing. Therefore, assumptions, encoded in the form of inductive bias, are a necessary component for generalization to unseen instances in classification on finite spaces.

When we consider the model of generalization used in statistical learning theory, we find that there "generalization" is simply defined to be performance over a test set drawn from the same distribution as the training set. Thus, the generalization error is often measured over instances that were seen in training in addition to the new instances. The possibility of shared instances allows for improved performance simply as a consequence of seeing more instances, which improves the chances that test instances will have been seen in training (for finite spaces, like those considered here). Thus, to consider the typical generalization error is to entangle two forms ability: memorization ability, for instances seen during training, and true generalization ability, which applies only to previously unseen instances. If our primary concern is minimizing risk under the instance distribution, then the blended form of generalization error is completely appropriate. However, discovering what allows machine learners to generalize beyond examples seen in training data requires that we separate the two abilities, and consider only performance on instances **not** seen during training. Thus, although the test set is assumed to be initially drawn from the same distribution as the training set, we exclude from our test set those instances contained in the training data. Doing so gives us a clearer statement of what can be gained from the processing of training data alone.

In addition to demonstrating the necessity of assumptions in classification learning, our results help us understand why this must be the case. Consider the space of possible hypotheses on the test data that are consistent with the training instances. Equally weighting every possible label for a given test instance means that every label is equally likely, and thus the proposed labels become independent of the instance features. No matter what relationships are found to hold between labels and features in the training set, the unbiased algorithm ignores these and does not require that the same relationships continue to hold within the test set. This allows for the possibility of unrepresentative training data sets, since relationships found there may not carry over to test data sets, at the cost of generalization ability. If, instead, we require that training data sets be representative, as is the case when the training and test data sets are drawn (i.i.d.) from the same instance distribution, then non-trivial relationships between labels and features of the training data set will continue to hold for test instances as well, allowing generalization to become possible. Thus, it is the dependence between features and labels that allows machine learning to work, and a lack of assumptions on the data generating process leads to independence, causing performance equivalent to random guessing. This underscores the importance of dependence between what is seen and what is unseen for successful machine learning.

Chapter 6

Statistical Learning Theory

VAPNIK'S statistical learning theory [81] has become something of a de facto paradigm for understanding modern machine learning, and has grown into a diverse field in its own right. We provide a brief overview of the concepts and goals of Vapnik's theory, and begin to explore the ways in which questions from statistical learning theory can be re-cast as search questions within our framework. The central concern of Vapnik's theory, empirical risk minimization, reduces to a search for models that minimize the empirical risk, making the necessary assumptions to ensure that the observed information resource feedback (empirical risk under a training dataset) is informative of the target (the true risk minimizer).

Vapnik's statistical learning theory reduces areas of machine learning to the problem of trying to minimize risk under some loss function, given a sample from an unknown (but fixed) distribution. The model for learning from examples can be described using three core components:

- generator of random vectors $P(x)$;
- supervisor, which returns an output vector y for each x, $P(y|x)$; and
- a learner that implements a set of functions $f_\alpha(x) = f(x, \alpha), \alpha \in \Lambda$.

The learning problem becomes choosing α so that $f_\alpha(x)$ predicts the response of the supervisor with the minimal amount of loss. Overall loss is measured using the risk functional on loss function L,

$$R(\alpha) = \int L(y, f_\alpha(x))dP(x, y).$$

Thus, the goal becomes finding an $\alpha_0 \in \Lambda$ that minimizes $R(\alpha)$ when $P(x, y)$ is unknown. Vapnik shows how this general setting can be applied to problems in classification, regression and density estimation, using appropriate loss functions, such as zero-one loss for classification and surprisal for density estimation.

The separate learning problems can be further abstracted to a common form as follows. Let $P(z)$ be defined on a space Z and assume a set of functions $Q_\alpha(z) = L(y, f_\alpha(x)), \alpha \in \Lambda$. The learning problem becomes minimizing $R(\alpha) = \int Q_\alpha(z)dP(z)$ for $\alpha \in \Lambda$ when $P(z)$ is unknown but an i.i.d. sample $z_1, \dots, z_\ell$ is given.

6.1 Empirical Risk Minimization

Define the empirical risk as

$$R_{emp}(\alpha) = \frac{1}{\ell}\sum_{i=1}^{\ell} Q_\alpha(z_i).$$

The *empirical risk minimization* (ERM) principle seeks to approximate $Q_{\alpha_0}(z_i)$, where α_0 minimizes the true risk, with $Q_{\alpha_\ell}(z_i)$, which minimizes the empirical risk. One then demonstrates conditions under which the true minimal risk is approached as a consequence of minimizing the empirical risk.

In the context of empirical risk minimization, two questions arise:

1. What are the necessary and sufficient conditions for the statistical consistency of the ERM principle?
2. At what rate does the sequence of minimized empirical risk values converge to the minimal actual risk?

The first question is answered by the theory of consistency for learning processes, which given appropriate statistical conditions, can show that $R(\alpha_\ell) \xrightarrow{p} R(\alpha_0)$, where $R(\alpha_\ell)$ is the risk and the α_ℓ are such that they minimize the empirical risk $R_{emp}(\alpha_\ell)$ for each $\ell = 1, 2, 3, \ldots$, and that $R_{emp}(\alpha_\ell) \xrightarrow{p} R(\alpha_0)$ as well. The second question is answered by the nonasymptotic theory of the rate of convergence, which uses concentration of measure to show at what rate the gap between the estimate and the optimal risk diminishes. The precise conditions for convergence are provided in [83] and [82], and demonstrate that restricting the complexity of a family of learners effectively constrains the ability of the empirical estimate of risk to differ considerably from the true risk. Here we will examine how the problem of empirical risk minimization can be viewed as a search problem within our framework.

6.1.1 Empirical Risk Minimization as Search

Expanding on the discussion given in Section 4.6, we review the translation of learning problem into search problem given there:

- $\Omega = \Lambda$;
- $T = \{\alpha : \alpha \in \Lambda, R(\alpha) - \min_{\alpha' \in \Lambda} R(\alpha') < \epsilon\}$;
- $F = \{z_1, \ldots, z_\ell\}$;
- $F(\emptyset) = \{z_1, \ldots, z_\ell\}$; and
- $F(\alpha) = R_{emp}(\alpha)$.

For a fixed sample size ℓ, ERM reduces to using the information gained from the empirical risk to locate an α sufficiently near the true minimal risk (and thus within the target). Two issues immediately present themselves: first, one must assume that the empirical risk will be sufficiently informative of the true risk (namely, a high degree of dependence between T and F), and second, one still needs to perform a search using that information resource F (the empirical risk). In [81], the assumption is made that the second search using F is always successful, so that the problem reduces to finding statistical conditions that guarantee F is informative of T. However, in many practical machine learning settings a successful search for the minimal empirical risk α is by no means assured. Performing this search forms the basis for much of applied machine learning and optimization research.

If we, like Vapnik, assume searches for empirical risk minimizers are always successful given an information resource, there is still the higher-level question of what statistical assumptions guarantee strong dependence between F and T. These questions are answered by statistical learning theory. In passing the baton to statistical learning theory our framework provides a natural scaffolding to anchor existing results in machine learning, providing a high-level overview that makes sense of (and ties together) low-level details. Our search framework naturally divides and highlights the primary concerns of applied and theoretical machine learning, showing how each contribute to overall success in learning.

It should also be noted that this form of learning theory is only concerned with reducing estimation error, namely, finding the best α within our class that reduces true risk. It says nothing about what the value of that minimal risk will be, whether small or large. By enlarging the class of functions considered one can possibly reduce the value of the minimal risk in the class, but must incur a cost of weakening the guarantees which rely on restricted classes of functions. Thus a trade-off exists between reducing the approximation error (i.e., the difference between the optimal risk for all functions and the best function in our class) and reducing the estimation error (reducing the error between our estimate and the best model possible within our class).

Chapter 7

Compression, Complexity and Learning

Plurality must never be posited without necessity

Sentences of Peter Lombard
William of Ockham

7.1 Machine Learning as a Communication Problem

As shown in Figure 7.1, machine learning problems can be represented as channel communication problems. Let each $z_i = (x_i, y_i)$ and assume we have some function $g(z)$ from which the data are generated according to a process. Given the output of that process (i.e., the training dataset), the job of a machine learning algorithm is to infer back the correct $\hat{g}$ approximation based on the training data. As should be obvious, unless we can place some constraints on how the data are related to the original function (namely, about characteristics of the communication channel), then reliable inference is impossible. These constraints may take the form of i.i.d. assumptions on the data generation, or parametric assumptions on how noise is added to labels, for example. Given some initial uncertainty regarding the true function g, it is hoped that $Z = \{z_1, \dots, z_n\}$ can reduce that uncertainty and lead to the recovery of g (or something close to it).

Given an output Z measurable in bits and a function g also representable using a finite number of bits, one can simultaneously look at the problem as a compression problem whenever $\text{bits}(Z) \ll \text{bits}(g)$. We will explore this in the context of binary classification, examining the link between generalization and compression and between compression and simplicity. It will be shown that while compression leads to good generalization guarantees, it does not do so for reasons that depend on simplicity in any intrinsic way, thus ruling out Occam's razor (quoted above) as a general explanation for why machine learning works.

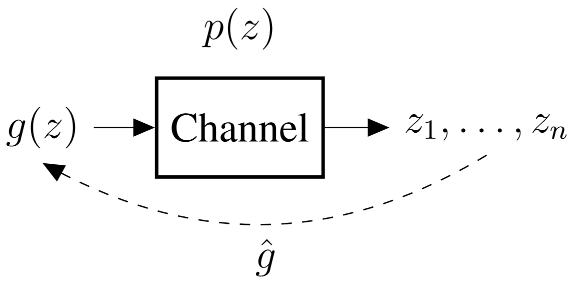


Figure 7.1: Machine learning as a channel communication problem.

7.2 Sample Compression and Binary Classification

Roughly thirty years ago, Littlestone and Warmuth [49] introduced a framework for sample compression and proved theorems showing that compressing samples into smaller subsamples results in good PAC generalization behavior. This line of research continues [12, 16, 59]. It is illustrative to look at a theorem of their original paper and see what makes the theorem work.

Theorem 8 (Littlestone and Warmuth [49], Thm 2.1)**.** *For any compression scheme reducing the training sample to a subsample of size k the error is larger than ϵ with probability (w.r.t. P^m) less than $\binom{m}{k}(1-\epsilon)^{m-k}$ when given an original sample of size $m \geq k$.*

Proof. It will suffice for our purposes to give a simple proof sketch that is faithful to the proof strategy followed by Littlestone and Warmuth. We have also slightly changed the wording of the theorem (removing references to a sample 'kernel' and replacing it with the explanation of what a kernel is in this context). The full proof is given in [49].

For any sample of size m, select k indices and extract the subsample residing at those indices, labeling that set S. This leaves a disjoint set V of all other $m-k$ samples. Using S as a parameterization set, assume the learning algorithm outputs a hypothesis that is consistent with all instances in V. Assuming the set V consists of independent samples which were not used in the construction of the hypothesis, the probability that all $m-k$ samples are consistent for a hypothesis with true generalization error greater than ϵ is no greater than $(1-\epsilon)^{m-k}$. Considering the union of all possible ways we could have selected a subset of k samples, namely $\binom{m}{k}$, and taking a union bound produces the final result. □

A similar proof of the same result is given in [59].

The fact that the result holds for *any* compression scheme should initially raise suspicion, since we have seen that any specific algorithm (whether built on compression, or otherwise) can only be successful in restricted settings. (See Theorem 3 in Chapter 5, for example, and also the discussions in [19, 70, 87, 93].) Thus, we investigate further. Even before seeing how the result is proven, we have a clue of how the proof will work based on the definition of *compression scheme*, which requires consistency with training examples. Combining that with the assumed i.i.d. dataset and PAC generalization loss (which differs from off-training-set error by allowing overlap between examples seen in training and testing) gives us a recipe for ensuring low generalization error. The basic idea is to split the data into a small training and a larger test set, and show that with high probability the hypothesis can only have good empirical loss on the independent test set if it also has good expected loss under the true distribution (from which the test set was drawn). Using a union bound to control for multiple hypothesis testing yields the final result. By only using a small portion of the data to train the model and keeping the test set V independent of the training set S, this ensures that empirical generalization performance correlates tightly with true generalization performance. Requiring the compression scheme used to have perfect empirical generalization performance ensures the agreement is with a small true generalization error value.

What we see "compression" here refers to how much of the data is available for testing versus training. It has nothing to do with complexity of the hypothesis class, once we condition on the size of the class. By compressing the sample down to a small subsample of training data, we are increasing the amount of testing data, which is what statistically powers the proof when taken together with the small number of possible hypotheses and assumption that our model class already contains at least one member capable of perfectly classifying all remaining training examples. Thus, complexity and compression are only indirectly involved in the bound; what is essential is having lots of independent testing data and assuming our class already contains a good model, while controlling for multiple-hypothesis testing.

7.3 The Proportion of Compressible Learnable Concepts

In the spirit of Theorem 8, we will prove a result relating compressibility and zero-one loss to the proportion of learnable concepts, for deterministic binary classification algorithms on finite instance spaces.

Theorem 9. *Let $\mathcal{X}$ be an instance space of size $|\mathcal{X}| = m \in \mathbb{N}$ and let Ω denote a concept space on $\mathcal{X}$ of size $|\Omega| = 2^m$. A fixed deterministic binary classification algorithm $\mathcal{A}$ with access to training datasets up to $m - b$ bits can achieve zero-one loss of ϵ on no more than*

$$(2^{m-b+1} - 1) \sum_{i=0}^{\epsilon m} \binom{m}{i}$$

concepts in Ω and the proportion of concepts learnable to within ϵ zero-one loss is upper bounded by

$$2^{-b+1} \left(\frac{e}{\epsilon}\right)^{\epsilon m}.$$

Proof. Let $\epsilon \in [0, 1]$ represent the fraction of errors made on classifying the instances of space $\mathcal{X}$ (as an enumerated list, not drawn from a distribution). Deterministic algorithms always map the same input to the same output, and since there are only $2^{m-b+1} - 1$ possible datasets of $m - b$ or fewer bits, such an algorithm can map to at most $2^{m-b+1} - 1$ output elements (the elements being concepts in Ω). Each output element has a Hamming sphere of at most $\sum_{i=0}^{\epsilon m} \binom{m}{i}$ nearby concepts with no more than proportion ϵ zero-one loss. Thus, given a collection of datasets each of fewer than $m - b$ bits, algorithm $\mathcal{A}$ can achieve less than ϵ zero-one loss for no more than $(2^{m-b+1} - 1) \sum_{i=1}^{\epsilon m} \binom{m}{i}$ possible concepts in Ω, the first result. Using the Sauer-Shelah inequality, we can upper bound the binomial sum by $(e/\epsilon)^{\epsilon m}$. Dividing by the total number of concepts in Ω yields the second. □

Although the first bound can be loose due to the possibility of overlapping Hamming spheres, in some cases it is tight. Thus, without making additional assumptions on the algorithm it cannot be generally improved upon. We examine one such special case next.

Example: Let $m = 1000$, $b = 999$ and $\epsilon = 0.001$. Since $m - b = 1$, we consider only single bit datasets (and the empty dataset). Each dataset can map to at most one concept in Ω, which for our example algorithm $\mathcal{A}$ will map 0 to the all zeros concept and 1 to the all ones concept, mapping the empty dataset to the concept that alternates between zero and one (i.e., $01010\ldots$). Given that $1000 \times 0.001 = 1$, the Hamming spheres for the concepts are all those that differ from the true concept by at most one bit, so that $\mathcal{A}$ can learn no more than 3,003 concepts in Ω (a proportion of roughly 2.8×10^{-298}) to within ϵ accuracy using only single-bit datasets. Our second looser bound puts the number at 10,873 total concepts with a proportion of 1.01×10^{-297}.

Example: Let $m = 100,000$, $b = 18,080$ and $\epsilon = 0.01$. Thus $m - b = 81,920$ bits (10 kb). A deterministic classification algorithm $\mathcal{A}$ can learn no more than 4.7×10^{-3009} of all concepts in Ω to within ϵ zero-one loss if it is limited to datasets of size 10 kb.

Neither of the bounds in Theorem 9 is very useful as one moves towards larger instance spaces and larger dataset sizes, due to the inability of most calculators to compute large exponents.

7.4 Compression and Generalization Bounds

As we saw in Section 7.2, there is a relatively long history of trying to understand machine learning through the lens of compression and proving bounds showing how compression leads to good generalization behavior. We will examine a modern version of this argument in the realm of empirical risk minimization, brought to my attention by Akshay Krishnamurthy. It uses a prefix-free code on a family of hypotheses to show that algorithms with short code lengths give tighter empirical risk minimization generalization bounds.

Theorem 10. *For a countably infinite class of functions $f \in \mathcal{F}$ under a prefix-free code and $\delta \in (0, 1]$,*

$$\Pr\left(\forall f \in \mathcal{F},\ |R_{emp}(f) - R(f)| \leq \sqrt{\frac{\log 2^{c(f)+1}/\delta}{2n}}\right) \geq 1 - \delta,$$

where $R_{emp}(f)$ is the empirical risk of function f on dataset $z_1, \ldots, z_n$, $R(f)$ is the true risk, and $c(f)$ is the code-length of the function under the prefix-free code.

Proof. For a fixed function f, Hoeffding's inequality gives with probability of at least $1 - \delta_f$ that $|R_{\text{emp}}(f) - R(f)| \leq \sqrt{\frac{\log 2/\delta_f}{2n}}$, showing that the sample average concentrates around the true mean at an $O(\sqrt{1/n})$ rate. To extend to a uniform bound for all functions in $\mathcal{F}$, set $\delta_f = \delta 2^{-c(f)}$, where $c(f)$ is the code-length of f under the prefix-free code. It follows that $\sum_{f \in \mathcal{F}} \delta_f \leq \delta$ by the Kraft-McMillan inequality for prefix-free codes. Then, defining $\mathcal{E}_{n,f} := \sqrt{\frac{\log 2^{c(f)+1}/\delta}{2n}}$, we have

$$
\begin{aligned}
\Pr\left(\forall f \in \mathcal{F}\ |R_{\text{emp}}(f) - R(f)| \le \mathcal{E}_{n,f}\right) &= \Pr\left(\bigcap_{f\in\mathcal{F}} \{|R_{\text{emp}}(f) - R(f)| \le \mathcal{E}_{n,f}\}\right) && (7.1)\\
&= 1 - \Pr\left(\bigcup_{f\in\mathcal{F}} \{|R_{\text{emp}}(f) - R(f)| > \mathcal{E}_{n,f}\}\right) && (7.2)\\
&\ge 1 - \sum_{f\in\mathcal{F}} \Pr\left(|R_{\text{emp}}(f) - R(f)| > \mathcal{E}_{n,f}\right) && (7.3)\\
&\ge 1 - \sum_{f\in\mathcal{F}} \delta_f && (7.4)\\
&\ge 1 - \delta. && (7.5)
\end{aligned}
$$

□

The result in Theorem 10 shows that one can get good bounds on how far the empirical risk deviates from the actual risk when using a family of functions drawn from a distribution and encoded using a prefix-free code. When a family of functions is countably infinite, it requires that most complex objects are given low probability under the distribution. We cannot assign large probabilities (short code-lengths) to every complex structure, given the infinite number of them. However, we can choose, if we wish, to assign large probabilities to every simple model, since there are relatively few of them. While this superficially ties function complexity to good generalization bounds, the next theorem will show that this is coincidental. "Complexity" is a free-rider, and what really matters is how the items are assigned codewords. Because complex objects outnumber simple ones, the encoding creates a correlation between complexity and deviation tightness. However, restricting ourselves to an immensely large but finite family of functions shows that the deviation tightness does not depend on complexity in any essential way.

Theorem 11. *For a finite class of functions $f \in \mathcal{F}$ under a prefix-free code where objects are assigned shorter codewords if they are* more *complex, and $\delta \in (0, 1]$,*

$$\Pr\left(\forall f \in \mathcal{F},\ |R_{emp}(f) - R(f)| \le \sqrt{\frac{\log 2^{c(f)+1}/\delta}{2n}}\right) \ge 1 - \delta,$$

where $R_{emp}(f)$ is the empirical risk of function f on dataset $z_1, \dots, z_n$, $R(f)$ is the true risk, and $c(f)$ is the code-length of the function under the prefix-free code.

Proof. The proof is identical to that of the previous theorem, since all steps apply equally to finite classes of functions under a prefix-free code, regardless of the code. □

In Theorem 11 we assume the prefix-free code is ranked in reverse order, such that the most complex functions have the shortest codewords, and the simplest objects are assigned the longest codewords. It follows that the complex functions achieve the best generalization bounds, demonstrating that the "complexity" argument was superficial. While sufficient under a specific assignment of codewords based on model simplicity, simplicity is not necessary for good generalization behavior, even in the case of countably infinite sets of functions. We show this next.

Theorem 12. *For any countably infinite class of functions $f \in \mathcal{F}$, $\delta \in (0, 1]$, any definition of "short code," and any ordering based on complexity (however defined), there exists a prefix-free code such that all short codes are assigned to complex functions,*

$$\Pr\left(\forall f \in \mathcal{F},\ |R_{emp}(f) - R(f)| \leq \sqrt{\frac{\log 2^{c(f)+1}/\delta}{2n}}\right) \geq 1 - \delta,$$

and the tightest bounds are not given by the simplest functions.

Proof. We will prove the existence of such a code by construction. For any countably infinite set $\mathcal{F}$, begin with a prefix-free code that assigns codeword lengths proportional to object complexity under the user-provided complexity ordering. Order the functions of $\mathcal{F}$ in ascending order by their code-lengths, breaking ties arbitrarily, and pick any N such that objects with order position greater than N have codes that are not "short" (under the given definition of short code) and are also complex (under the notion of complexity used in the user-defined ordering). Take the first $2N$ functions and reverse their order, such that the $2N$th function has the shortest codeword, and the simplest object is assigned the codeword formerly belonging to the $2N$th object. Since all short codewords fall within the first N positions and all objects that were formerly between N and $2N$ are complex, in the new code all short codewords belong to complex objects. □

Theorem 12 presents a case where tight ERM deviation bounds arise in spite of complexity of the functions in their final prefix-free ordering, not because of it. At best, complexity constraints can give rise to sufficient conditions for attaining tight generalization bounds, but as the previous two results have shown, complexity notions are not necessary for such bounds. The notion of simplicity and the notion of compression are somewhat orthogonal; this again suggests that Occam-style ordering of models is not a general explanation for why machine learning works. The next section provides additional evidence.

7.5 Occam's Razor and ML

Shalizi [73] has argued against the use of Occam's Razor and compression-based generalization bounds as a final justification for machine learning. Conceding that for compression-based bounds like Theorem 10, "[t]he shorter the description, the tighter the bound on the generalization error," he points out:

> What actually makes the proofs go is the fact that there are not very many binary strings of length m for small m. Finding a classifier with a short description means that you have searched over a small space. It's the restriction of the search space which really does all the work. It's not the *simplicity* of the rules which matters, but the fact that simple rules are scarce in the space of all possible rules. If confines oneself to elaborate, baroque rules, but sticks to a *particular style* of baroque elaboration, one obtains the same effect.
>
> For example, suppose we have n data points for a binary classification problem. Enumerate the rules in some order, say lexicographic. Now consider the set of rules whose serial numbers are exactly m^{m^m}, with m being any prime number less than

or equal to n. By the prime number theorem, there are approximately $n/\ln(n)$ such prime numbers. Suppose one of these rules correctly classifies all the data. The likelihood of it doing so if its actual error rate were q is $(1-q)^n$. By a union bound, the probability that the actual error rate is q or more is $n(1-q)^n/\ln(n)$. Notice that no matter how low an error rate I demand, I can achieve it with arbitrarily high confidence by taking n sufficiently large. But, notice, most of the rules involves here have enormous serial numbers, and so must be very long and complicated programs. That doesn't matter, because there still just aren't many of them. In fact, this argument would still work if I replaced the prime less than or equal to n with a *random* set, provided the density of the random set was $1/\ln(n)$.

Mitchell also discusses the use of a form of Occam's Razor in machine learning [54]. He states:

> One argument is that because there are fewer short hypotheses than long ones (based on straightforward combinatorial arguments), it is less likely that one will find a short hypothesis that coincidentally fits the training data. In contrast there are often many very complex hypotheses that fit the current training data but fail to generalize correctly to subsequent data. Consider decision tree hypotheses, for example. There are many more 500-node decision trees than 5-node decision trees. Given a small set of 20 training examples, we might expect to be able to find many 500-node decision trees consistent with these, whereas we would be more surprised if a 5-node decision tree could perfectly fit this data. We might therefore believe the 5-node tree is less likely to be a statistical coincidence and prefer this hypothesis over the 500-node hypothesis.
>
> Upon closer examination, it turns out there is a major difficulty with the above argument. By the same reasoning we could have argued that one should prefer decision trees containing exactly 17 leaf nodes with 11 nonleaf nodes, that use the decision attribute A_1 at the root, and test attributes A_2 through A_{11}, in numerical order. There are relatively few such trees, and we might argue (by the same reasoning as above) that our a priori chance of finding one consistent with an arbitrary set of data is therefore small. The difficulty here is that there are very many small sets of hypotheses that one can define-most of them rather arcane. Why should we believe that the small set of hypotheses consisting of decision trees with *short descriptions* should be any more relevant than the multitude of other small sets of hypotheses that we might define?

Thus, his arguments anticipate Shalizi's and are consistent with the counterexamples presented here.

Domingos' critique of Occam's razor arguments in machine learning [19] is equally devastating. Discussing the role of PAC generalization guarantees in the context i.i.d. datasets, he notes:

> For the present purposes, the results can be summarized thus. Suppose that the generalization error of a hypothesis is greater than ϵ. Then the probability that the hypothesis is correct on m independent examples is smaller than $(1-\epsilon)^m$. If there are $|H|$ hypotheses in the hypothesis class considered by a learner, the probability

> that at least one is correct on all m training examples is smaller than $|H|(1-\epsilon)^m$, since the probability of a disjunction is smaller than the sum of the probabilities of the disjuncts. Thus, if a model with zero training-set error is found within a sufficiently small set of models, it is likely to have low generalization error. This model, however, could be arbitrarily complex. The only connection of this result to Occam's razor is provided by the information-theoretic notion that, if a set of models is small, its members can be distinguished by short codes. But this in no way endorses, say, decision trees with fewer nodes over trees with many. By this result, a decision tree with one million nodes extracted from a set of ten such trees is preferable to one with ten nodes extracted from a set of a million, given the same training-set error.
>
> Put another way, the results in Blumer et al. [11] only say that if we select a sufficiently small set of models prior to looking at the data, and by good fortune one of those models closely agrees with the data, we can be confident that it will also do well on future data. The theoretical results give no guidance as to how to select that set of models.

This agrees with the observation that it is the small set sizes and assumption that the sets contain at least one good hypothesis that does the heavy-lifting in such proofs; it is not the inherent complexity of the members of the sets themselves.

Domingos defines two forms of Occam's razor, one favoring simplicity as an end goal in itself (the first razor), and the second favoring simpler models because of the belief it will lead to lower generalization error. He argues that use of the first razor is justified, but the second is problematic. After reviewing several empirical studies showing how complex models, including ensemble methods, often outperform simpler models in practice, Domingos concludes, "All of this evidence supports the conclusion that not only is the second razor not true in general; it is also typically false in the types of domains [that knowledge discovery / data-mining] has been applied to."

7.6 The Minimum Description Length Principle

The Minimum Description Length (MDL) Principle is a way of looking at machine learning as a type of compression problem [32, 34]. In its most general formulation, it casts statistical inference as the problem of trying to find regularities in data, and identifies 'finding regularities' with compression ability. Thus, the goal of MDL systems are to find the hypotheses that compress the data the most. While the MDL principle can be applied to problems in classification, regression, and other areas of machine learning, most of its formal developments have been in the area of *model selection* [32], where a researcher uses data to select among various hypotheses.

In [32], Grünwald begins with an example of polynomial curve-fitting, reproduced in Figure 7.2, to explain what MDL seeks to accomplish. He states that the first image on the left (the straight line) is too simple to fit the data, whereas the center polynomial, while fitting each point much better, is overly complex and will not likely fit a new set of data drawn from the original distribution. In the final image on the right (i.e., the third-degree polynomial) he sees the ideal trade-off between simplicity and fit, and claims that this curve would likely fit new data better

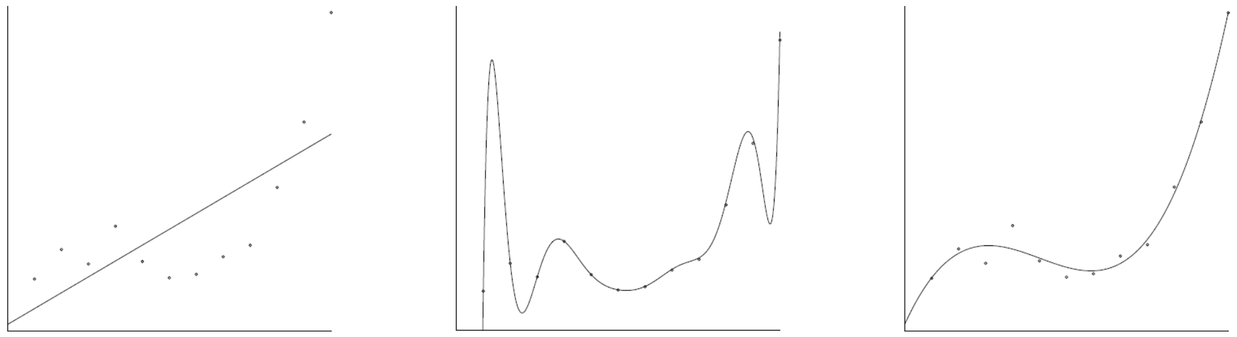


Figure 7.2: Polynomial curve fitting. A simple, a complex and 3rd degree polynomial. Image reproduced from [32].

than either of the other polynomials shown. For Grünwald, such an "intuition is confirmed by numerous experiments on real-world data from a broad variety of sources [66, 67, 84]." Although such examples and intuitions sound plausible in certain contexts (e.g., when a simple function is corrupted by noise, leading to data that are more complex than the true underlying function), they do not seem to be universally valid [19, 87]. In fact, our earlier theoretical results suggest that *no* bias can be universally valid unless the universe has graciously constrained itself to producing a select subset of possible problems.

Returning to the polynomial-fitting example, one can argue that the primary problem of the high-degree curve is not that it is 'complex,' but that it places the curve in regions where no data demands it. In other words, while it explains what is there (the data points), it also explains what isn't there (the regions where long stretches of line are not supported by any data point). In contrast, the straightforward technique of connecting each neighboring data point by a line segment leads to a jagged line perfectly fitting all points. Such a 'curve' loses smoothness properties and would be highly complex (requiring on the order of n parameters to define the curve fitting n data points), but remains close to the true curve as long as the data points themselves are representative of that curve. Thus, we have a highly-complex, flexible model (growing on order n, the size of the data), that also intuitively will give good generalization performance. The difference with our jagged model is that while it adequately explains what is there, it minimizes the amount of curve unsupported by observations, thus not explaining what *isn't* there. This is closer in spirit to E.T. Jayne's *maximum entropy principle* [43], which seeks to avoid (as much as possible) making assumptions beyond what is supported by the data. While we must *always* make assumptions beyond what is seen in the data in order to generalize (see Theorem 7, for example), we can be careful to not make more assumptions than are actually necessary.

7.6.1 MDL: Balancing Fit and Simplicity

MDL, in its more refined forms, seeks to minimize a composite loss, containing a term that accounts for model fit given a model class (typically a conditional log-loss / surprisal term) and a term accounting for the complexity of the model class (similar in spirit to a VC dimension). This can be represented as a model relative loss for data D, namely

$$Q(D; \mathcal{M}) = -\log P(D|\hat{\theta}_{\mathcal{M}}(D)) + \text{COMP}_{|D|}(\mathcal{M})$$

where $\hat{\theta}_{\mathcal{M}}(D)$ identifies the model within class $\mathcal{M}$ maximizing the likelihood of the data, and $\text{COMP}_{|D|}(\mathcal{M})$ measures the 'richness' and 'flexibility' (i.e., complexity) of the model class $\mathcal{M}$. MDL seeks to manage the trade-off between fit and complexity via minimizing the quantity $Q(D; \mathcal{M})$. Part of the historical challenge of MDL theory has been how to define a code that measures the model complexity $\text{COMP}_{|D|}(\mathcal{M})$ in a non-arbitrary way, and Grünwald [32] gives examples of codes formulated to do so in a principled and sensible way, such as by using *universal codes* that minimize minimax regret. It has been shown that in certain contexts MDL methods are statistically consistent [97], although not being formulated with that goal explicitly in mind.

7.6.2 MDL as the Explanation for Learning

While MDL methods are powerful, elegant, and have many other attractive properties, they cannot universally explain successful machine learning. If they could, then simply seeking to compress regularities in data would always correspond with successful learning, and more broadly, Occam-style justifications for learning would not admit the easy counterexamples we have found in this chapter. As evidence against the first point, we see that in some situations MDL procedures can perform poorly in practice. Grünwald explains [32]:

> However, there is a class of problems where MDL is problematic in a more fundamental sense. Namely, if none of the distributions under consideration represents the data generating machinery very well, then both MDL and Bayesian inference may sometimes do a bad job in finding the best approximation within this class of not-so-good hypotheses. This has been observed in practice.

This suboptimal performance under misspecification has been explored in [33].

Thus, since seeking to minimize the description of data is not sufficient to guarantee successful learning, we see that MDL (and other biases towards simplicity) cannot be the grand unifying principle explaining machine learning. At best, MDL represents a bias towards less complex models that succeeds in many real-world situations and has several advantages, but being a bias nonetheless, it cannot serve as a universal explanation for success in learning.

7.6.3 Reasons for Compressing Regularities

Setting aside the question of whether MDL presents a universally valid justification for statistical inference, one can see that compression of data into simple models can be a worthwhile pursuit for several other reasons have little to do with generalization or 'truth.'

First, in big data situations, it is often preferable to reduce or eliminate the data instances into a small parametric model that is fast to use and requires little storage space. To the degree that a learning procedure can reduce the large amount of data into a small model, we make gains in efficiency and usability. Thus, this goal is valid irrespective of whether simpler models are more often correct. Even in situations where they are not and we knowingly incur some loss by compressing down our data, the savings may be large enough to justify the loss in fidelity. Rate distortion theory [6] and lossy compression schemes manage this exact trade-off.

Second, being simple creatures ourselves, we often prefer simple models for interpretability reasons. They are easier to understand and follow.

Third, smoothness and regularity imply both compressibility and dependence, where dependence exists between nearby points in space or time. When a function is smooth it means that if we know something about the value of a point at x we also know something about the likely values of points at $x + \epsilon$ that are nearby. Thus, smoothness induces exploitable dependence for learning. Simple functions are relatively easy to learn, and since we must begin with some bias, a bias towards simplicity (and learnability) is not a bad bias to start with. It may be demonstrably wrong, but so will every other bias in certain situations.

Fourth, in situations where noise is present, the noise will typically cause the data to become more complex than the generating signal, which justifies regularized methods and biasing them towards models that are simpler than the data themselves appear. When our assumptions and biases align to what holds in the actual world, our methods perform well in practice. It follows that in noisy situations a bias towards simplicity is reasonable and can lead to good empirical performance (though not always – see [70] and accompanying discussion in Section 2.4.2).

Lastly, in the small data case where there are few data samples, it may be better to prefer simple models (with fewer parameters) over more complex ones, since there will typically be more data points per parameter when fitting the simpler model. Because of limitations on statistical power, a bias towards simpler models becomes justified in such situations.

Part III

Examples and Applications

SETTING out to answer the question "Why does machine learning work?," we learned that we could reduce machine learning to a form of search and investigate the dual question of what makes searches successful. Our theoretical results showed us that for any fixed algorithm, relatively few possible problems are greatly favorable for the algorithm. The subset of problems which are favorable require a degree of dependence between what is observed and what is not, between labels and features, datasets and concepts. Without strong dependence between the target set and the external information resource, the probability of a successful search remains negligible. Successful learning thus becomes a search for exploitable dependence. We can use this view of machine learning to uncover exploitable structure in problem domains and inspire new learning algorithms.

In the chapters that follow, we will apply our dependence-first view of learning to two real-world problem areas, showing how dependence can lead to greater-than-chance learning performance. This confirms the practical utility of our framework for machine learning and applied research. It should be noted, however, that the algorithms developed are not *derived* explicitly from the formal theorems proven, but are instead guided by the "dependence-first" view of learning that naturally results from considering machine learning within our framework.

We begin by exploring a set of simple examples showing direct application of our formal results and their consequences for a few problem domains, then explore two application areas in depth. These problem areas are unsupervised time-series segmentation and hyperparameter learning. For the first task, we leverage temporal dependence in our time series data to achieve improved segmentation results, and for the second task, we leverage spatial dependence. Both types of dependence flow from smoothness and regularity within the problem domain, a common source of exploitable structure within machine learning.

Chapter 8

Examples[1]

8.1 Landmark Security

IMAGINE a fictional world where a sophisticated criminal organization plots to attack an unspecified landmark within a city. Due to the complexity of the attack and methods of infiltration, the group is forced to construct a plan relying on the coordinated actions of several interdependent agents, of which the failure of any one would cause the collapse of the entire plot. As a member of the city's security team, you must allocate finite resources to protect the many important locations within the city. Although you know the attack is imminent, your sources have not indicated which location, of the hundreds possible, will be hit; your lack of manpower forces you to make assumptions about target likelihoods. You know you can foil the plot if you stop even one enemy agent. Because of this, you seek to maximize the odds of capturing an agent by placing vigilant security forces at the strategic locations throughout the city. Allocating more security to a given location increases surveillance there, raising the probability a conspirator will be found if operating nearby. Unsure of your decisions, you allocate based on your best information, but continue to second-guess yourself.

With this picture in mind, we can analyze the scenario through the lens of algorithmic search. Our external information resource is the pertinent intelligence data, mined through surveillance. We begin with background knowledge (represented in $F(\emptyset)$), used to make the primary security force placements. Team members on the ground communicate updates back to central headquarters, such as suspicious activity, which are the $F(\omega)$ evaluations used to update the internal information state. Each resource allocated is a query, and manpower constraints limit the number of queries available. Doing more with fewer officers is better, so the hope is to maximize the per-officer probability of stopping the attack.

Our results tell us a few things. First, a fixed strategy can only work well in a limited number of situations. There is little or no hope of a problem being a good match for your strategy if the problem arises independently of it (Theorem 2). So reliable intelligence becomes key. The better correlated the intelligence reports are with the actual plot, the better a strategy can perform (Theorem 4). However, even for a fixed search problem with reliable external information resource

[1]This chapter reproduces content from Montañez, "The Famine of Forte: Few Search Problems Greatly Favor Your Algorithm" (arXiv 2016)

there is no guarantee of success, if the strategy is chosen poorly; the proportion of good strategies for a fixed problem is no better than the proportion of good problems for a fixed algorithm (Theorem 3). Thus, domain knowledge is crucial in choosing either. Without side-information to guide the match between search strategy and search problem, the expected probability of success is dismal in target-sparse situations.

8.2 One-Size-Fits-All Fitness Functions

Theorem 2 gives us an upper bound on the proportion of favorable search problems, but what happens when we have a single, fixed information resource, such as a single fitness function? A natural question to ask is for how many target locations can such a fitness function be useful. More precisely, for a given search algorithm, for what proportion of search space locations can the fitness function raise the expected probability of success, assuming a target element happens to be located at one of those spots?

Applying Theorem 2 with $|T| = 1$, we find that a fitness function can significantly raise the probability of locating target elements placed on, at most, $1/q_{\text{min}}$ search space elements. We see this as follows. Since $|T| = 1$ and the fitness function is fixed, each search problem maps to exactly one element of Ω, giving $|\Omega|$ possible search problems. The number of q_{min}-favorable search problems is upper-bounded by

$$|\Omega|\frac{p}{q_{\text{min}}} = |\Omega|\frac{|T|/|\Omega|}{q_{\text{min}}} = |\Omega|\frac{1/|\Omega|}{q_{\text{min}}} = \frac{1}{q_{\text{min}}}.$$

Because this expression is independent of the size of the search space, the number of elements for which a fitness function can strongly raise the probability of success remains fixed even as the size of the search space increases. Thus, for very large search spaces the proportion of favored locations effectively vanishes. There can exist no single fitness function that is strongly favorable for many elements simultaneously, and thus no "one-size-fits-all" fitness function.

8.3 Binary Classification

As we saw previously, we can directly represent binary classification problems within our framework. Given the components of our framework, a typical binary classification problem can be reduced to a search problem in our framework as follows:

- $\mathcal{A}$ - classification algorithm, such as logistic regression.
- Ω - space of possible concepts over an instance space.
- T - Set of all hypotheses with less than 10% classification error on test set, for example.
- F - set of training examples, i.e., $(x_1, y_1), \ldots, (x_N, y_N)$.
    - $F(\emptyset)$ - full set of training data.
    - $F(h_i)$ - loss on training data for concept h_i.
- (Ω, T, F) - binary classification learning task.

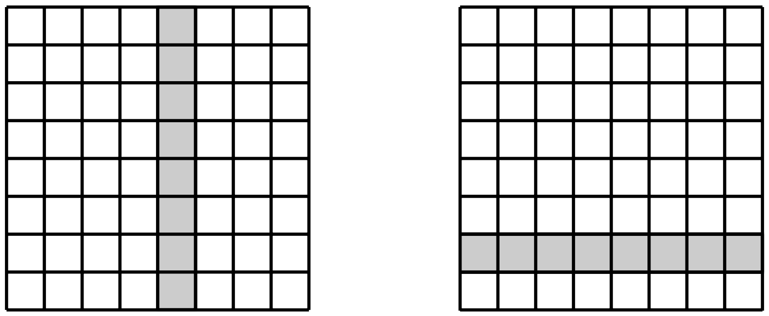


Figure 8.1: Possible axis-aligned target set examples.

The space of possible binary concepts is Ω, with the true concept being an element in that space. In our example, let $|\Omega| = 2^{100}$. The target set consists of the set of all concepts in that space that (1) are consistent with the training data (which we will assume all are), and (2) differ from the truth in at most 10% of positions on the generalization held-out dataset. Thus, $|T| = \sum_{i=0}^{10} \binom{100}{i}$. Let us assume the marginal distribution on T is uniform, which isn't necessary but simplifies the calculation. The external information resource F is the set of training examples. The algorithm uses the training examples (given by $F(\emptyset)$) to produce a distribution over the space of concepts; for deterministic algorithms, this is a degenerate distribution on exactly one element. A single query is then taken (i.e., a concept is output), and we assess the probability of success for the single query. The chance of outputting a concept with at least 90% generalization accuracy is thus no greater than $\frac{I(T;F)+1}{I_\Omega} \approx \frac{I(T;F)}{I_\Omega} \leq \frac{I(T;F)}{59}$. The denominator is the information cost of specifying at least one element of the target set and the numerator represents the information resources available for doing so. When the mutual information meets (or exceeds) that cost, success can be ensured for any algorithm perfectly mining the available mutual information. When noise reduces the mutual information below the information cost, the probability of success becomes strictly bounded in proportion to that ratio.

8.4 One Equation Examples

Using the equation from Theorem 5, we can see how information resources are transformed into probability of success in learning. We will do so for three simple examples.

8.4.1 Example: Strong Target Structure

Let Ω be an 8×8 grid and $|T| = k = 8$. Assume that target sets are always axis-aligned either vertically or horizontally, so that they fill either a column or row of Ω as in Figure 8.1. Choosing a target set is therefore equivalent to choosing a row or column. Let the distribution on target sets P_T be uniform on rows and columns (i.e., the target set can be in any row or column, uniformly at random). Lastly, we will assume the information resource contains no information concerning the target set (i.e., $I(T;F) = 0$) and that the information leakage I_L is zero. Under these conditions we expect any algorithm to have a probability of success for a single query equivalent to choosing elements uniformly at random (since it has no external information to guide it), which should give a probability of success equal to $8/64 = 0.125$. We will see that Equation 5.43 gives precisely this answer.

Under our assumptions, we compute the remaining components:

- $I_\Omega = -\log_2 8/64 = 3$;
- $H(T) = 4$ (since it takes one bit to choose between rows or columns, and three bits to specify the row or column);
- $D(P_T \| \mathcal{U}_T) = \log_2 \binom{64}{8} - H(T) = 32.04 - 4 = 28.04$;
- $H(Z|X) = -1/8 \log_2 1/8 - 7/8 \log_2 7/8 = 0.5436$;
- $\mathbb{E}_X \left[D(P_{T|Z=0,X} \| \mathcal{U}_{T|Z=0,X}) \right] = \log_2 \binom{63}{8} - \log_2 1/14 = 28.04$;
- $\mathbb{E}_X \left[D(P_{T|Z=1,X} \| \mathcal{U}_{T|Z=1,X}) \right] = \log_2 \binom{63}{7} - 1 = 28.04$ (since $H(T|Z = 1, X) = 1$, because we have two options if we know an element of T: either it is on the row or column of X); and
- $Cr = \log_2 7/8 - 28.04 = -28.233$.

Thus, we get

$$\Pr(X \in T; \mathcal{A}) = \frac{I(T;F) - I_L + D(P_T \| \mathcal{U}_T) + H(Z|X) + Cr}{I_\Omega + \mathbb{E}_X \left[D(P_{T|Z=1,X} \| \mathcal{U}_{T|Z=1,X}) \right] + Cr} \tag{8.1}$$

$$= \frac{0 - 0 + 28.04 + 0.5436 - 28.233}{3 + 28.04 - 28.233} \tag{8.2}$$

$$= 0.125. \tag{8.3}$$

8.4.2 Example: Minimal Target and Positive Dependence

Let Ω again be the same 8×8 grid and let $k = |T| = 1$, so that the target set consists of a single element. Let $P_T = \mathcal{U}_T$ and assume $I_L = 0$. Under these conditions, we have the following:

- $I_\Omega = -\log_2 1/64 = 6$;
- $H(T) = 6$;
- $D(P_T \| \mathcal{U}_T) = 0$;
- $\mathbb{E}_X \left[D(P_{T|Z=0,X} \| \mathcal{U}_{T|Z=0,X}) \right] = \log_2 \binom{63}{1} - H(T|Z = 0, X) = \log_2 63 - H(T|Z = 0, X)$;
- $\mathbb{E}_X \left[D(P_{T|Z=1,X} \| \mathcal{U}_{T|Z=1,X}) \right] = \log_2 \binom{63}{0} - H(T|Z = 1, X) = \log_2 1 - 0 = 0$; and
- $Cr = \log_2 63/64 - \log_2 63 + H(T|Z = 0, X) = H(T|Z = 0, X) - 6$.

Plugging this into our formula, we obtain

$$\Pr(X \in T; \mathcal{A}) = \frac{I(T;F) - I_L + D(P_T \| \mathcal{U}_T) + H(Z|X) + Cr}{I_\Omega + \mathbb{E}_X \left[D(P_{T|Z=1,X} \| \mathcal{U}_{T|Z=1,X}) \right] + Cr} \tag{8.4}$$

$$= \frac{I(T;F) - 0 + 0 + H(Z|X) + H(T|Z = 0, X) - 6}{6 + 0 + H(T|Z = 0, X) - 6} \tag{8.5}$$

$$= \frac{I(T;F) - 6 + H(Z|X) + H(T|Z = 0, X)}{H(T|Z = 0, X)}. \tag{8.6}$$

If we want to upper bound this quantity, we can use the fact that $H(Z|X) \leq 1$ and note that the ratio is monotonically increasing in $H(T|Z=0,X)$, for which we have $H(T|Z=0,X) \leq H(T) = 6$. It follows that,

$$\Pr(X \in T; \mathcal{A}) \leq \frac{I(T;F)+1}{6}. \tag{8.7}$$

When $I(T;F) = 2$, then the probability of success can be no greater than 0.5; when $I(T;F) = 0.1$, the maximum drops to 0.183. This upper bound coincides with that given in Theorem 4, which is much easier to use. An exact form simpler to use than Equation 5.43 is suggested by Equation 5.50,

$$\Pr(X \in T; \mathcal{A}) = \frac{I(T;F) - I_L - H(T) + H(T|Z=0,X) + H(Z|X)}{H(T|Z=0,X) - H(T|Z=1,X)}.$$

This form is easier to compute and requires fewer components to reason about. It may prove more useful in practice.

8.4.3 Example: Target Avoidance Algorithm

Since mutual information captures both correlation and anti-correlation, we want to ensure our equation for successful learning accurately reflects what happens when an algorithm uses available mutual information to avoid hitting a target. Given that the mutual information available could be used for either finding or avoiding the target, we expect the equation to account for this type of contrary behavior.

To begin, let us again have an 8×8 grid as our search space Ω with a single element target set T, so that $k = |T| = 1$. Let the target set be chosen uniformly at random on Ω, so that $P_T = \mathcal{U}_T$. Assume the information has full information concerning the target set, so that $I(T;F) = H(T)$. Furthermore, assume that the search algorithm actively avoids the target in the following way, for $\epsilon > 0$:

- with probability $(1-\epsilon)$ choosing the square directly north (if possible), or directly south if the target is on the top row; and
- with probability ϵ choosing the target element.

Given this process, we know that the algorithm will succeed with probability exactly ϵ, which can be chosen as small as one likes (as long as positive). Thus, we expect $\Pr(X \in T; \mathcal{A}) = \epsilon$.

Let us compute the terms we need for Equation 5.43:

- $H(T) = -\log_2 1/64 = 6$;
- $I(T;F) = H(T) = 6$;
- $I_\Omega = -\log_2 1/64 = 6$;
- $D(P_T \| \mathcal{U}_T) = 0$;
- $\mathbb{E}_X\left[D(P_{T|Z=0,X} \| \mathcal{U}_{T|Z=0,X})\right] = \log_2 \binom{63}{1} - H(T|Z=0,X) = \log_2 63 - H(T|Z=0,X)$;
- $\mathbb{E}_X\left[D(P_{T|Z=1,X} \| \mathcal{U}_{T|Z=1,X})\right] = \log_2 \binom{63}{0} - H(T|Z=1,X) = \log_2 1 - 0 = 0$;
- $Cr = \log_2(1 - k/|\Omega|) - \mathbb{E}_X\left[D(P_{T|Z=0,X} \| \mathcal{U}_{T|Z=0,X})\right] = H(T|Z=0,X) - 6$;

- $H(T|Z=1,X)=0$; and
- $H(T|Z=0,X)=\frac{1}{4}-\frac{\epsilon}{8}$;

The computation for $H(T|Z=0,X)$ can be seen as follows. First, we compute $P(X)$ by marginalizing over the Z values and computing $P(X|Z)P(Z)$. The conditional probability will change depending on which row X is on, given our search strategy. Taking this into account, we find that $P(X)$ equals $1/4-\epsilon/8$ if on the second row, $\epsilon/8$ if on the last row, and $1/8$ otherwise. Next, we notice that for all rows other than the second, knowing X and that $Z=0$ allows you to uniquely pinpoint T, reducing its conditional entropy to zero. For X elements chosen from the second row, there are two equally weighted possibilities for T, either above or below, thus giving one bit of uncertainty. Multiply this by the marginal probability of choosing an element in the second row, $1/4-\epsilon/8$, and we obtain that value.

We need to also compute I_L in this case. We have

$$\begin{aligned} I_L &= I(T;F)-I(T;X) && (8.8)\\ &= H(T)-[H(T)-H(T|X)] && (8.9)\\ &= H(T|X). && (8.10)\end{aligned}$$

Using the chain rule for entropy two ways on $H(T,Z|X)$ and remembering that $H(Z|T,X)=0$ by definition, that $Z:=\mathbb{1}_{X\in T}$, and that $P(Z=1)=\epsilon$, we obtain

$$\begin{aligned} I_L &= H(T|X) && (8.11)\\ &= H(T|Z,X)+H(Z|X) && (8.12)\\ &= P(Z=0)H(T|Z=0,X)+P(Z=1)H(T|Z=1,X)+H(Z|X) && (8.13)\\ &= (1-\epsilon)H(T|Z=0,X)+H(Z|X). && (8.14)\end{aligned}$$

Thus,

$$\begin{aligned} \Pr(X\in T;\mathcal{A}) &= \frac{I(T;F)-I_L+D(P_T\|\mathcal{U}_T)+H(Z|X)+Cr}{I_\Omega+\mathbb{E}_X\left[D(P_{T|Z=1,X}\|\mathcal{U}_{T|Z=1,X})\right]+Cr} && (8.15)\\ &= \frac{6-(1-\epsilon)H(T|Z=0,X)+0+H(T|Z=0,X)-6}{6+0+H(T|Z=0,X)-6} && (8.16)\\ &= \frac{\epsilon H(T|Z=0,X)}{H(T|Z=0,X)} && (8.17)\\ &= \epsilon. && (8.18)\end{aligned}$$

Chapter 9

Unsupervised Segmentation of Time Series[1]

TIME series have become ubiquitous in many areas of science and technology. Sophisticated methods are often required to model their dynamical behavior. Furthermore, their dynamical behavior can change over time in systematic ways. For example, in the case of multivariate accelerometer data taken from human subjects, one set of dynamical behaviors may hold while the subject is walking, only to change to a new regime once the subject begins running. Gaining knowledge of these *change points* can help in the modeling task, since given a segmentation of the time series, one can learn a more precise model for each regime. Change point detection algorithms [45, 50, 65, 96] have been proposed to determine the location of systematic changes in the time series, while Hidden Markov Models (HMM) can both determine where regime (state) changes occur, and also model the behavior of each state.

One crucial observation in many real-world systems, natural and man-made, is the behavior changes are typically infrequent; that is, the system takes some (typically unknown) time before it changes its behavior to that of a new state. In our earlier example, it would be unlikely for a person to rapidly switch between walking and running, making the durations of different activities over time relatively long and highly variable. We refer to this as the *inertial property*, in reference to a physical property of matter that ensures it continues along a fixed course unless acted upon by an external force. Unfortunately, classical HMMs trained to maximize the likelihood of data can result in abnormally high rates of state transitioning, as this often increases the likelihood of the data under the model, leading to false positives when detecting change points, as is seen in Figure 9.1.

The inertial property of real-world time series represents a potentially exploitable source of dependence: temporal consistency. Since states are not likely to rapidly fluctuate, this increases dependence between neighboring points, since knowledge of current state reveals the likely next state, thus reducing uncertainty for the next point. In this thesis, we seek to operationalize this insight to produce a system capable of exploiting temporal consistency. We do so by introducing temporal regularization for HMMs, forming Inertial Hidden Markov Models [58]. These models are able to successfully segment multivariate time series from real-world accelerometer data and synthetic datasets, in an unsupervised manner, improving on state-of-the-art Bayesian nonparametric sticky hierarchical Dirichlet process hidden Markov models (HDP-HMMs) [27, 89]

[1]This chapter reproduces content from Montañez et al., "Inertial Hidden Markov Models: Modeling Change in Multivariate Time Series" (AAAI-2015)

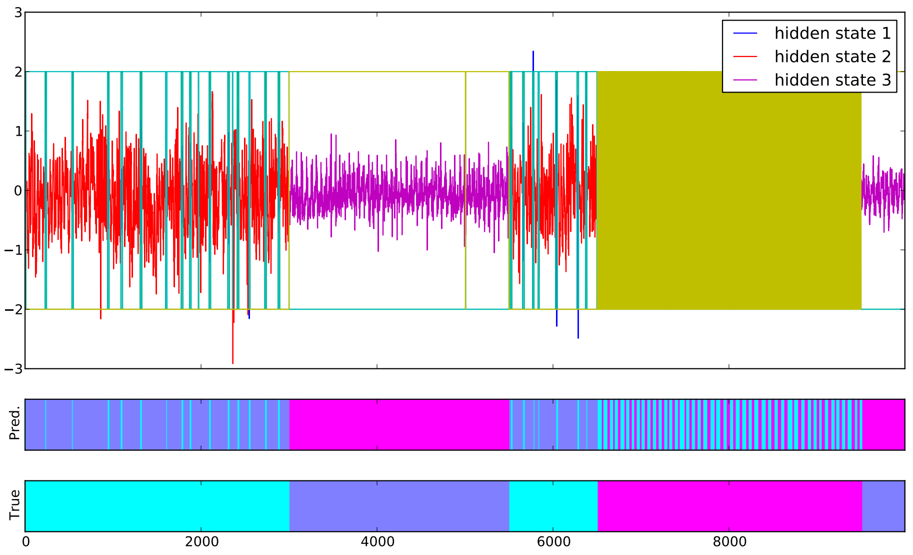


Figure 9.1: Classical HMM segmentation of human accelerometer activity data.

as well as classical HMMs. This application demonstrates the practical utility of the dependence-first view of learning inspired by our search framework.

9.1 Inertial HMM

Inertial HMMs are built on the foundations of classical HMMs [63], but we impose two additional criteria on our models. First, we seek models with a small number of latent states, $K \ll T$, and second, we desire state transition sequences with the inertial property, as defined previously, that lack rapid transitioning back and forth between states.

The above desiderata must be externally imposed, since simply maximizing the likelihood of the data will result in $K = T$ (i.e., each sample corresponds a unique state/distribution), and in general we may have rapid transitions among states in classical HMMs. The first issue is addressed by choosing the number of states in advance as is typically done for hidden Markov models [63]. For the second, we directly alter the probabilistic form of our model to include a parameterized regularization that reduces the likelihood of transitioning between different latent states. We thus begin a standard K-state HMM with Gaussian emission densities. HMMs trained by expectation maximization (locally) maximize the likelihood of the data, but typically do not guarantee slow inertial transitioning among states. Although the number of states must be specified in advance, no other parameters need to be given, as the remaining parameters are all estimated directly from the data.

We now present two methods for enforcing slow inertial transitioning among states. Both methods alter the complete data likelihood of the data, which have the effect of altering the resulting transition matrix update equations.

9.2 Maximum A Posteriori (MAP) Regularized HMM

Following [30], we alter the standard HMM to include a Dirichlet prior on the transition probability matrix, such that transitions out-of-state are penalized by some regularization factor. A Dirichlet prior on the transition matrix $\mathbf{A}$, for the jth row, has the form

$$p(A_j; \eta) \propto \prod_{k=1}^{K} A_{jk}^{\eta_{jk}-1}$$

where the η_{jk} are free parameters and A_{jk} is the transition probability from state j to state k. The posterior joint density over $\mathbf{X}$ and $\mathbf{Z}$ becomes

$$P(\mathbf{X}, \mathbf{Z}; \theta, \eta) \propto \left[\prod_{j=1}^{K} \prod_{k=1}^{K} A_{jk}^{\eta_{jk}-1} \right] P(\mathbf{X}, \mathbf{Z} \mid \mathbf{A}; \theta)$$

and the log-likelihood is

$$\begin{aligned} \ell(\mathbf{X}, \mathbf{Z}; \theta, \eta) \propto & \sum_{j=1}^{K} \sum_{k=1}^{K} (\eta_{jk} - 1) \log A_{jk} + \log P(\mathbf{z}_1; \theta) \\ & + \sum_{t=1}^{T} \log P(\mathbf{x}_t | \mathbf{z}_t; \theta) + \sum_{t=2}^{T} \log P(\mathbf{z}_t | \mathbf{z}_{t-1}; \theta). \end{aligned}$$

MAP estimation is then used in the M-step of the expectation maximization (EM) algorithm to update the transition probability matrix. Maximizing, with appropriate Lagrange multiplier constraints, we obtain the update equation for the transition matrix,

$$A_{jk} = \frac{(\eta_{jk} - 1) + \sum_{t=2}^{T} \xi(z_{(t-1)j}, z_{tk})}{\sum_{i=1}^{K} (\eta_{ji} - 1) + \sum_{i=1}^{K} \sum_{t=2}^{T} \xi(z_{(t-1)j}, z_{ti})} \tag{9.1}$$

where $\xi(z_{(t-1)j}, z_{tk}) := \mathbb{E}[z_{(t-1)j} z_{tk}]$.

Given our prior, we can control the probability of self-transitions among states, but this method requires that we choose a set of K^2 parameters for the Dirichlet prior. However, since we are solely concerned about increasing the probability of self-transitions, we can reduce these parameters to a single parameter λ governing the amplification of self-transitions. We therefore define $\eta_{jk} = 1$ when $j \neq k$ and $\eta_{kk} = \lambda \geq 1$ otherwise, and the transition update equation becomes

$$A_{jk} = \frac{(\lambda - 1)\mathbb{1}(j = k) + \sum_{t=2}^{T} \xi(z_{(t-1)j}, z_{tk})}{(\lambda - 1) + \sum_{i=1}^{K} \sum_{t=2}^{T} \xi(z_{(t-1)j}, z_{ti})} \tag{9.2}$$

where $\mathbb{1}(\cdot)$ denotes the indicator function.

9.3 Inertial Regularization via Pseudo-observations

Alternatively, we can alter the HMM likelihood function to include a latent binary random variable, V, indicating that a self-transition was chosen at random from among all transitions, according to some distribution. Thus, we view the transitions as being partitioned into two sets, self-transitions and non-self-transitions, and we draw a member of the self-transition set according to a Bernoulli distribution governed by parameter p. Given a latent state sequence $\mathbf{Z}$, with transitions chosen according to transition matrix $\mathbf{A}$, we define p as a function of both $\mathbf{Z}$ and $\mathbf{A}$. We would like p to have two properties: 1) it should increase with increasing $\sum_k A_{kk}$ (probability of self-transitions) and 2) it should increase as the number of self-transitions in $\mathbf{Z}$ increases. This will allow us to encourage self-transitions as a simple consequence of maximizing the likelihood of our observations.

We begin with a version of p based on a penalization constant $0 < \epsilon < 1$ that scales appropriately with the number of self-transitions. If we raise ϵ to a large positive power, the resulting p will decrease. Thus, we define p as ϵ raised to the number of non-self-transitions, M, in the state transition sequence, so that the probability of selecting a self-transition increases as M decreases. Using the fact that $M = (T-1) - \sum_{t=2}^{T}\sum_{k=1}^{K} z_{(t-1)k}z_{tk}$, we obtain

$$\begin{aligned} p = \epsilon^{M} &= \epsilon^{\sum_{t=2}^{T} 1 - \sum_{t=2}^{T}\sum_{k=1}^{K} z_{(t-1)k}z_{tk}} \\ &= \epsilon^{\sum_{t=2}^{T}\sum_{k=1}^{K} z_{(t-1)k} - \sum_{t=2}^{T}\sum_{k=1}^{K} z_{(t-1)k}z_{tk}} \\ &= \prod_{t=2}^{T}\prod_{k=1}^{K} \epsilon^{z_{(t-1)k} - z_{(t-1)k}z_{tk}}. \end{aligned} \tag{9.3}$$

Since ϵ is arbitrary, we choose $\epsilon = A_{kk}$, to allow p to scale appropriately with increasing probability of self-transition. We therefore arrive at

$$p = \prod_{t=2}^{T}\prod_{k=1}^{K} A_{kk}^{z_{(t-1)k} - z_{(t-1)k}z_{tk}}.$$

Thus, we define p as a computable function of $\mathbf{Z}$ and $\mathbf{A}$. Defining p in this deterministic manner is equivalent to choosing the parameter value from a degenerate probability distribution that places a single point mass at the value computed, allowing us to easily obtain a posterior distribution on V. Furthermore, we see that the function increases as the number of self-transitions increases, since $A_{kk} \leq 1$ for all k, and p will generally increase as $\sum_k A_{kk}$ increases. Thus, we obtain a parameter $p \in (0, 1]$ that satisfies all our desiderata. With p in hand, we say that V is drawn according to the Bernoulli distribution, Bern(p), and we observe $V = 1$ (i.e., a member of the self-transition set was chosen).

To gain greater control over the strength of regularization, let λ be a positive integer and $\mathbf{V}$ be an λ-length sequence of pseudo-observations, drawn i.i.d. according to Bern(p). Since $P(V = 1|\mathbf{Z}; \mathbf{A}) = p$, we have

$$P(\mathbf{V} = \mathbf{1}|\mathbf{Z}; \mathbf{A}) = \left[\prod_{t=2}^{T}\prod_{k=1}^{K} A_{kk}^{z_{(t-1)k} - z_{(t-1)k}z_{tk}}\right]^{\lambda}$$

where $\mathbf{1}$ denotes the all-ones sequence of length λ.

Noting that $\mathbf{V}$ is conditionally independent of $\mathbf{X}$ given the latent state sequence $\mathbf{Z}$, we maximize (with respect to A_{jk}) the expected (with respect to $\mathbf{Z}$) joint log-density over $\mathbf{X}$, $\mathbf{V}$, and $\mathbf{Z}$ parameterized by $\theta = \{\pi, \mathbf{A}, \phi\}$, which are the start-state probabilities, state transition matrix and emission parameters, respectively. Using appropriate Lagrange multipliers, we obtain the regularized maximum likelihood estimate for A_{jk}:

$$A_{jk} = \frac{B_{j,k,T} + \mathbb{1}(j = k)C_{j,k,T}}{\sum_{i=1}^{K} B_{j,i,T} + C_{j,j,T}} \tag{9.4}$$

where $\mathbb{1}(\cdot)$ denotes the indicator function, $\gamma(z_{tk}) := \mathbb{E}[z_{tk}]$ and

$$\begin{aligned} B_{j,k,T} &= \sum_{t=2}^{T} \xi(z_{(t-1)j}, z_{tk}), \\ C_{j,k,T} &= \lambda \left[\sum_{t=2}^{T} [\gamma(z_{(t-1)k}) - \xi(z_{(t-1)j}, z_{tk})] \right]. \end{aligned} \tag{9.5}$$

The forward-backward algorithm can then be used for efficient computation of the γ and ξ values, as in standard HMMs [10].

Ignoring normalization, we see that

$$A_{jk} \propto \begin{cases} B_{j,k,T} + C_{j,j,T} & \text{if } j = k \\ B_{j,k,T} & \text{otherwise.} \end{cases}$$

Examining the $C_{j,j,T}$ term (i.e., Equation (9.5)), we see that λ is a multiplier of additional mass contributions for self-transitions, where the contributions are the difference between $\gamma(z_{(t-1)j})$ and $\xi(z_{(t-1)j}, z_{tj})$. These two quantities represent, respectively, the expectation of being in a state j at time $t - 1$ and the expectation of remaining there in the next time step. The larger λ or the larger the difference between arriving at a state and remaining there, the greater the additional mass given to self-transition.

9.3.1 Parameter Modifications

Scale-Free Regularization

In Equation 9.2, the strength of the regularization diminishes with growing T, so that asymptotically the regularized estimate and unregularized estimate become equivalent. While this is desirable in many contexts, maintaining a consistent strength of inertial regularization becomes important with time series of increasing length, as is the case with online learning methods. Figure 9.2 shows a regularized segmentation of human accelerometer data (discussed later in the Experiments section), where the regularization is strong enough to provide good segmentation. If we then increase the number of data points in each section by a factor of ten while keeping the same regularization parameter setting, we see that the regularization is no longer strong enough, as is shown in Figure 9.3. Thus, the λ parameter is sensitive to the size of the time series.

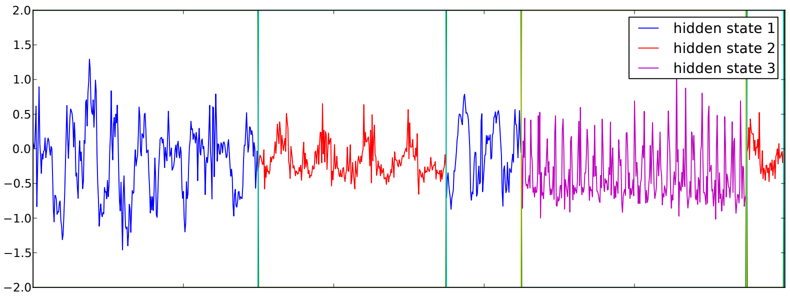


Figure 9.2: Human activities accelerometer data, short sequence. Vertical partitions correspond to changes of state.

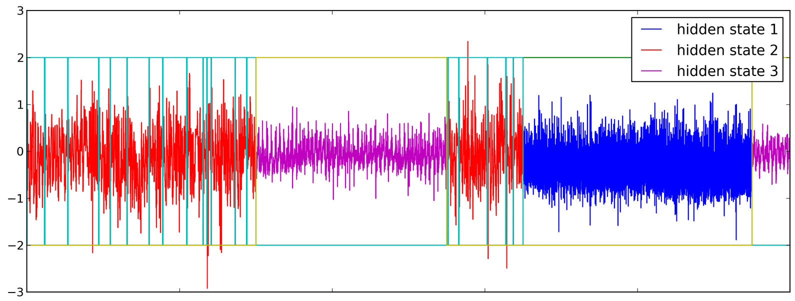


Figure 9.3: The long sequence human activities accelerometer data using regularization parameter from short sequence.

We desire models where the regularization strength is scale-free, having roughly the same strength regardless of how the time series grows. To achieve this, we define the λ parameter to scale with the number of transitions, namely $\lambda = (T-1)^{\zeta}$, and our scale-free update equation becomes

$$A_{jk} = \frac{((T-1)^{\zeta} - 1)\mathbb{1}(j = k) + \sum_{t=2}^{T} \xi(z_{(t-1)j}, z_{tk})}{((T-1)^{\zeta} - 1) + \sum_{i=1}^{K} \sum_{t=2}^{T} \xi(z_{(t-1)j}, z_{ti})}. \tag{9.6}$$

This preserves the effect of regularization as T increases, and ζ becomes our new regularization parameter, controlling the strength of the regularization. For consistency, we also re-parameterize Equation (9.5) using $\lambda = (T-1)^{\zeta}$.

Towards Parameter-Free Regularization

Although our methods require specifying the strength of regularization in advance, we can often avoid this requirement. For example, cross-validation provides a robust method for automated parameter selection when labeled data is available. Furthermore, even in the absence of labeled data all hope is not lost; if one can make the assumption that most of the segment lengths are of roughly the same order-of-magnitude scale then automatic tuning of the regularization parameter remains possible, as follows.

We first define a range of possible regularization parameter values (such as $\lambda \in [0, 75]$), and perform a search on this interval for a value that gives *sufficient regularization*. Sufficient regularization is defined with respect to the Gini ratio [31, 88], which is a measure of statistical dispersion often used to quantify income inequality. For a collection of observed segment lengths $L = \{l_1, \ldots, l_m\}$, given in ascending order, the Gini ratio is estimated by

$$G(L) = 1 - \frac{2}{m-1}\left(m - \frac{\sum_{i=1}^{m} i l_i}{\sum_{i=1}^{m} l_i}\right).$$

We assume that the true segmentation has a Gini ratio less than one-half, which corresponds to having more equality among segment lengths than not. One can perform a binary search on the search interval to find the smallest ζ parameter for which the Gini ratio is at least one-half. This increases the time complexity by a factor of $O(\log_2(R/\epsilon))$, where R is the range of the parameter space and ϵ is the stopping precision for the binary search.

9.4 Experiments

We perform two segmentation tasks on synthetic and real multivariate time series data, using our scale- and parameter-free regularized inertial HMMs. For comparison, we present the results of applying a standard K-state hidden Markov model as well as the sticky HDP-HMM of [27]. We performed all tasks in an unsupervised manner, with state labels being used only for evaluation.

9.4.1 Datasets

The first (synthetic) multivariate dataset was generated using a two-state HMM with 3D Gaussian emissions, with transition matrix

$$\mathbf{A} = \begin{pmatrix} 0.9995 & 0.0005 \\ 0.0005 & 0.9995 \end{pmatrix},$$

equal start probabilities and emission parameters $\mu_1 = (-1,-1,-1)^\top$, $\mu_2 = (1,1,1)^\top$, $\mathbf{\Sigma}_1 = \mathbf{\Sigma}_2 = \text{diag}(3)$. Using this model, we generated one hundred time series consisting of ten-thousand time points each. Figure 9.4 shows an example time series from this synthetic dataset.

The second dataset was generated from real-world forty-five dimensional human accelerometer data, recorded for users performing five different activities, namely, playing basketball, rowing, jumping, ascending stairs and walking in a parking lot [2]. The data were recorded from a single subject using five Xsens MTx$^{\text{TM}}$ units attached to the torso, arms and legs. Each unit had nine sensors, which recorded accelerometer (X, Y, Z) data, gyroscope (X, Y, Z) data and magnetometer (X, Y, Z) data, for a total of forty-five signals at each time point.

We generated one hundred multivariate time series from the underlying dataset, with varying activities (latent states) and varying number of segments. To generate these sets, we first uniformly chose the number of segments, between two and twenty. Then, for each segment, we chose an activity uniformly at random from among the five possible, and selected a uniformly random segment length proportion. The selected number of corresponding time points were extracted from the activity, rescaled to zero mean and unit variance, and appended to the output sequence. The final output sequence was truncated to ten thousand time points, or discarded if the sequence contained fewer than ten thousand points or fewer than two distinct activities.

Additionally, prospective time series were rejected if they caused numerical instability issues for the algorithms tested. The process was repeated to generate one hundred such multivariate time series of ten thousand time ticks each, with varying number of segments, activities and segment lengths. An example data sequence is shown in Figure 9.5 and the distribution of the time series according to number of activities and segments is shown in Figure 9.6.

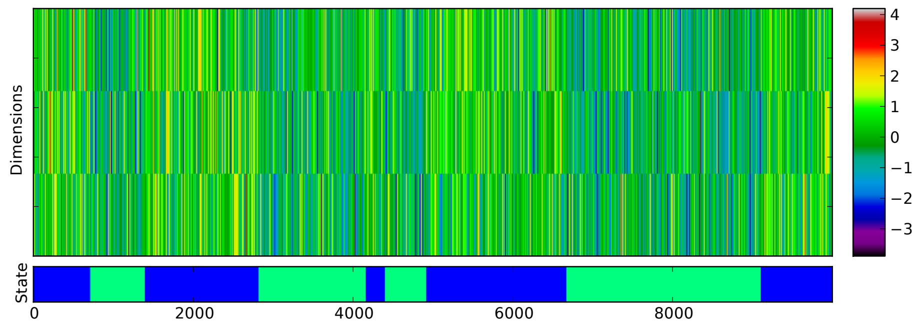


Figure 9.4: Synthetic data example. Generated from two-state HMM with 3D Gaussian emissions and strong self-transitions.

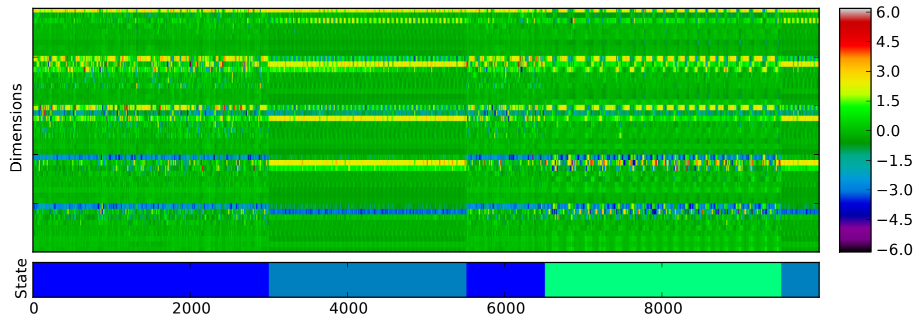


Figure 9.5: Human activities accelerometer data. Three state, 45-dimensional.

9.4.2 Experimental Methodology

We compared performance of four methods on the two datasets described in the previous section: a standard K-state hidden Markov model, the sticky HDP-HMM and both inertial HMM variants. All HMMs were equipped with Gaussian emission models with full covariance matrices. The task was treated as a multi-class classification problem, measuring the minimum zero-one loss under all possible permutations of output labels, to accommodate permuted mappings of the true labels. We measured the normalized variation of information (**VOI**) [53] between the predicted state sequence and true state sequence, which is an information metric capturing the distance between two partitionings of a sequence. We also considered the ratio of predicted number of segments to true number of segments (**SNR**), which gives a sense of whether a method over- or under-segments data, and the absolute segment number ratio (**ASNR**), which is defined as

$$\text{ASNR} = \max(S_t, S_p)/\min(S_t, S_p),$$

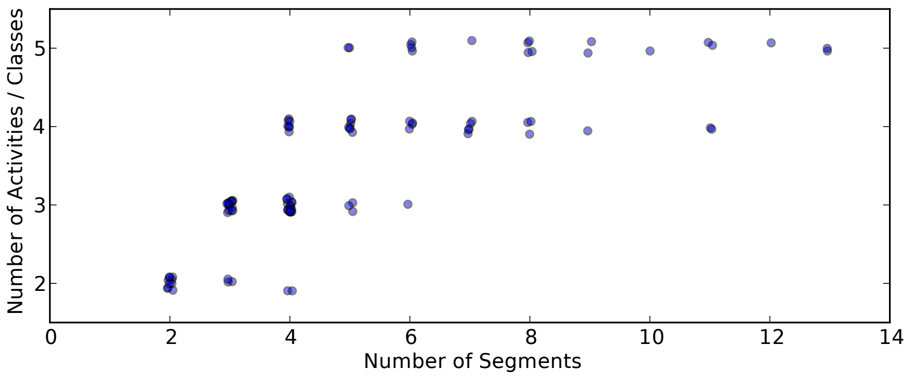


Figure 9.6: Distribution of Accelerometer Time Series Data.

where S_t is the true number of segments in the sequence and S_p is the predicted number of segments, and quantifies how much a segmentation method diverges from the ground truth in terms of relative factor of segments. Lastly, we tracked the number of segments difference (**SND**) between the predicted segmentation and true segmentation and how many segmentations we done perfectly (**Per.**), giving the correct states at all correct positions.

Parameter selection for the inertial HMM methods was done using the automated parameter selection procedure described in the Parameter Modifications section. For faster evaluation, we ran the automated parameter selection process on ten randomly drawn examples, averaged the final ζ parameter value, and used the fixed value for all trials. The final ζ parameters are shown in Tables 9.1 and 9.2.

To evaluate the sticky HDP-HMM, we used the publicly available HDP-HMM toolbox for MATLAB, with default settings for the priors [26]. The Gaussian emission model with normal inverse Wishart (NIW) prior was used, and the truncation level L for each example was set to the true number of states, in fairness for comparing with the HMM methods developed here, which are also given the true number of states. The "stickiness" κ parameter was chosen in a data-driven manner by testing values of $\kappa = 0.001$, 0.01, 0.1, 1, 5, 10, 50, 100, 250, 500, 750 and 1000 for best performance over ten randomly selected examples each. The mean performance of the 500th Gibbs sample of ten trials was then taken for each parameter setting, and the best κ was empirically chosen. For the synthetic dataset, a final value of $\kappa = 10$ was chosen by this method. For the real human accelerometer data, a value of $\kappa = 100$ provided the best accuracy and relatively strong variation of information performance. These values were used for evaluation on each entire dataset, respectively.

To evaluate the HDP-HMM, we performed five trials on each example in the test dataset, measuring performance of the 1000th Gibbs sample for each trial. The mean performance was then computed for the trials, and the average of all one hundred test examples was recorded.

9.4.3 Synthetic Data Results

As seen in Table 9.1, the MAP regularized HMM had the strongest performance, with top scores on all metrics. The inertial pseudo-observation HMM also had strong performance, with extremely high accuracy and low variation of information. The standard HMM suffered from

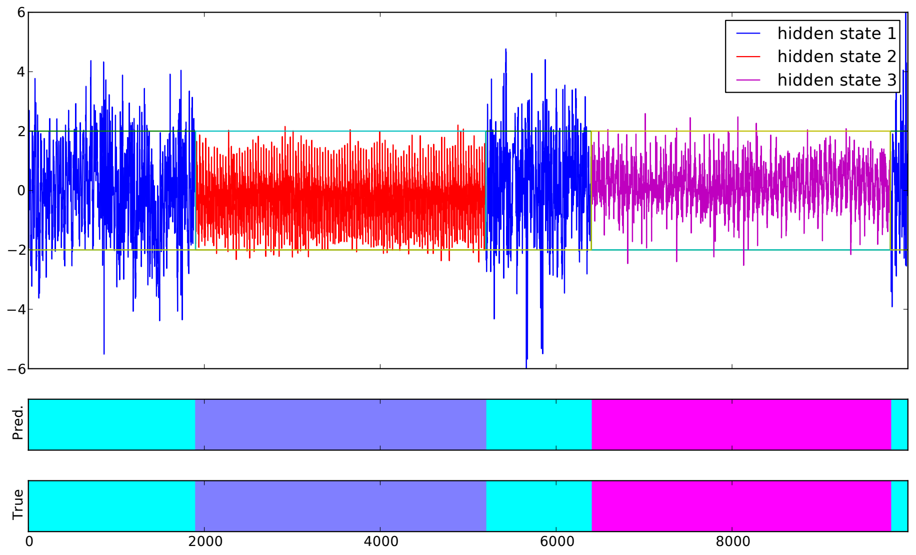


Figure 9.7: Example segmentation of human activities accelerometer data using inertial (MAP) HMM. Only first dimension shown.

Table 9.1: Results from quantitative evaluation on 3D synthetic data. Statistical significance is computed with respect to MAP results.

Method	**Acc.**	**SNR**	**ASNR**	**SND**	**VOI**	**Per.**
HDP-HMM ($\kappa = 10$)	0.85*	0.59*	3.50*	2.79*	0.56*	0/100
Standard HMM	0.87*	172.20*	172.20*	765.91*	0.62*	0/100
MAP HMM ($\zeta = 2.3$)	**0.99**	**0.96**	**1.13**	**0.51**	**0.07**	**2/100**
PsO HMM ($\zeta = 8.2$)	**0.99**	0.87‡	1.43‡	1.15*	0.14†	1/100

Acc. = Average Accuracy (value of 1.0 is best)
SNR = Average Segment Number Ratio (value of 1.0 is best)
ASNR = Average Absolute Segment Number Ratio (value of 1.0 is best)
SND = Average Segment Number Difference (value of 0.0 is best)
VOI = Average Normalized Variation of Information (value of 0.0 is best)
Per. = Total number of perfect/correct segmentations

paired t-test: $\dagger < \alpha = .05$, $\ddagger < \alpha = .01$, $* < \alpha = .001$

over-segmentation of the data (as reflected in the high SNR, ASNR, and SND scores), while the sticky HDP-HMM tended to under-segment the data. All methods were able to achieve fairly high accuracy.

9.4.4 Human Activities Accelerometer Data Results

Table 9.2: Results from real 45D human accelerometer data.

Method	Acc.	SNR	ASNR	SND	VOI	Per.
HDP-HMM ($\kappa = 100$)	0.60*	0.75‡	4.68*	5.03‡	0.95*	0/100*
Standard HMM	0.79*	134.59*	134.59*	584.16*	0.38*	9/100*
MAP HMM ($\zeta = 33.5$)	**0.94**	1.28	1.43	2.62	**0.14**	**48/100**
PsO HMM ($\zeta = 49.0$)	**0.94**	**1.03**†	**1.29**	**1.29**	0.15	**48/100**

paired t-test: † $< \alpha = .05$, ‡ $< \alpha = .01$, * $< \alpha = .001$

Results from the human accelerometer dataset are shown in Table 9.2. Both the MAP HMM and inertial pseudo-observation HMM achieved large gains in performance over the standard HMM model, with average accuracy of 94%. Furthermore, the number of segments was close to correct on average, with a value near one in both the absolute (ASNR) and simple (SNR) ratio case. The average normalized variation of information (VOI) was low for both the MAP and pseudo-observation methods. Figure 9.7 shows an example segmentation for the MAP HMM, displaying a single dimension of the multivariate time series for clarity.

In comparison, the standard hidden Markov model performed poorly, strongly over-segmenting the sequences in many cases. Even more striking was the improvement over the sticky HDP-HMM, which had an average normalized variation of information near 1 (i.e., no correlation between the predicted and the true segment labels). The method tended to *under*-segment the data, often collapsing to a single uniform output state, reflected in the SNR having a value below one, and may struggle with moderate dimensional data, as related by Fox and Sudderth through private correspondence. Moreover, the poor performance on this dataset likely results from a strong dependence on Bayesian tuning parameters. The sticky HDP-HMM suffers from slow mixing rates as the dimensionality increases, and computation time explodes, being roughly cubic in the dimension. As a result, the one hundred test examples took several days of computation time to complete, whereas the inertial HMM methods took a few hours.

9.5 Discussion

Our results demonstrate the effectiveness of inertial regularization on HMMs for behavior change modeling in multivariate time series. Although derived in two independent ways, the MAP regularized and pseudo-observation inertial regularized HMM converge on a similar maximum likelihood update equation, and thus, had similar performance.

The human activity task highlighted an issue with using standard HMMs for segmentation of time series with infrequent state changes, namely, over-segmentation. Incorporating regularization for state transitions provides a simple solution to this problem. Since our methods rely on changing a single update equation for a standard HMM learning method, they can be easily incorporated into HMM learning libraries with minimal effort. This ease-of-implementation gives a strong advantage over existing persistent-state HMM methods, such as the sticky HDP-HMM framework.

While the sticky HDP-HMM performed moderately well on the low-dimensional synthetic dataset, the default parameters produced poor performance on the real-world accelerometer data. It remains possible that different settings of hyperparameters may improve performance, but the cost of a combinatorial search through hyperparameter space combined with lengthy computation time prohibits an exhaustive exploration. The results, at minimum, show a strong dependence on hyperparameter settings for acceptable performance. In contrast, the inertial HMM methods make use of a simple heuristic for automatically selecting the strength parameter ζ, which resulted in excellent performance on both datasets without the need for hand-tuning several hyperparameters. Although the sticky HDP-HMM has poor performance on the two segmentation tasks, there exist tasks for which it may be a better choice (e.g., when the correct number of states is unknown).

9.6 Related Work

Hidden Markov models for sequential data have enjoyed a long history, gaining popularity as a result of the widely influential tutorial by Rabiner [63]. Specific to the work presented here, the mechanics of applying regularization priors to HMMs was detailed in [30], for both transition and emission parameters, with the goal of solving estimation issues arising from sparse data. Our work introduces the use of regularization for enforcing state persistence, an application not considered in the original. Neukirchen and Rigoll [62] studied the use of regularization in HMMs for reducing parameter overfitting of emission distributions due to insufficient training data, again without an emphasis on inertial transitioning between states. Similarly, Johnson [44] proposed using Dirichlet priors on multinomial hidden Markov models as a means of enforcing sparse emission distributions.

Fox *et al.* [27] developed a Bayesian sticky HMM to provide inertial state persistence. They presented a method capable of learning a hidden Markov model without specifying the number of states or regularization-strength beforehand, using a hierarchical Dirichlet process and truncated Gibbs sampling. As discussed, their method requires a more complex approach to learning the model and specification of several hyperparameters for the Bayesian priors along with a truncation limit. In contrast, our models only require the specification of two parameters, K and ζ, whereas the sticky HDP-HMM requires analogous truncation level L and κ parameters to be chosen, in addition to the hyperparameters on the model priors.

Chapter 10

Automated Hyperparameter Tuning [1]

THE rapid expansion of machine learning methods in recent decades has created a common question for non-expert end users: "What hyperparameter settings should I use for my algorithm?" Even the simplest algorithms often require the tuning of one or more hyperparamters, which can yield significant effects on performance [7, 14]. However, knowing which settings to use for which dataset and algorithm has remained something of an "art," relying on the implicit knowledge of practitioners in the field. Because employing machine learning methods should not require a PhD in data science, researchers have recently begun to investigate automated hyperparameter tuning methods, with marked success ([7, 8, 9, 25, 39, 76, 80, 86]). These methods have been successfully applied to a wide range of problems, including neural networks and deep belief network hyperparameter tuning [8, 9], thus showing promise for automatically tuning the large numbers of hyperparameters required by deep learning architectures. We extend this body of research by improving on the state-of-the-art in automated hyperparameter tuning, guided by our search view of machine learning. Because dependence makes searches successful, we begin by first investigating which dependencies hold and then appropriately biasing our search procedure to exploit the dependencies found, giving improved performance in our original learning problem.

We introduce a new sequential model-based, gradient-free optimization algorithm, Kernel Density Optimization (KDO), which biases the search in two ways. First, it assumes strong spatial consistency of the search space, such that nearby points in the space have similar function values, and second, it assumes that sets of randomly chosen points from the space will have function evaluations that follow a roughly unimodal, approximately Gaussian distribution. These assumptions hold for several real-world hyperparameter optimization problems on UCI datasets using three different learning methods (gradient boosted trees, regularized logistic regression, and averaged perceptrons), allowing our method to significantly improve on the state-of-the-art SMAC method. Thus, we gain increased empirical performance by appropriately biasing our algorithm based on our knowledge of dependencies.

[1]This chapter reproduces content from Montañez and Finley, "Kernel Density Optimization"*(In Prep.)*

10.1 Theoretical Considerations

Given that sequential hyperparameter optimization is a literal search through a space of hyperparameter configurations, our results are directly applicable. The search space Ω consists of all the possible hyperparameter configurations (appropriately discretized in the case of numerical hyperparameters). The target set T is determined by the particular learning algorithm the configurations are applied to, the performance metric used, and the level of performance desired. Let S denote a set of points sampled from the space, and let the information gained from the sample become the external information resource f. Given that resource, we have the following theorem:

Theorem 13. *Given a search algorithm $\mathcal{A}$, a finite discrete hyperparameter configuration space Ω, a set S of points sampled from that search space, and information resource f that is a function of S, let $\Omega' := \Omega \setminus S$, $\tau_k = \{T \mid T \subseteq \Omega', |T| = k \in \mathbb{N}\}$, and $\tau_{k,q_{min}} = \{T \mid T \in \tau_k, q(T,f) \geq q_{min}\}$, where $q(T,f)$ is the expected per-query probability of success for algorithm $\mathcal{A}$ under T and f. Then,*

$$\frac{|\tau_{k,q_{min}}|}{|\tau_k|} \leq \frac{p'}{q_{min}}$$

where $p' = k/|\Omega'|$

The proof follows directly from Theorem 2.

The proportion of possible hyperparameter target sets giving an expected probability of success q_{min} or more is minuscule when $k \ll |\Omega'|$. If we have no additional information beyond that gained from the points S, we have no justifiable basis for expecting a successful search. Thus, we must make some assumptions concerning the relationship of the points sampled to the remaining points in Ω'. We can do so by either assuming structure on the search space, such that spatial coordinates becomes informative, or by making an assumption on the process by which S was sampled, so that the sample is representative of the space in quantifiable ways. These assumptions allow f to become informative of the target set T, leading to exploitable dependence. Thus we see the need for inductive bias in hyperparameter optimization [55], which hints at a strategy for creating more effective hyperparameter optimization algorithms (i.e., through exploitation of spatial structure). We adopt this strategy for KDO.

10.2 KDO: Motivation

Imagine a hyperparameter space where all numerical axes have been rescaled to the [0,1] range, leaving us with a hyperparameter cube to explore. The goal of hyperparameter optimization is to locate (and sample from) regions of the cube with good empirical performance (when the parameter settings are applied to real algorithms on real tasks). Grid-based search methods poorly explore the space, since on a single axis, they repeatedly sample the same points [8]. Uniform random sampling improves the subspace coverage, but samples equally from promising and unpromising regions. A better method would be to characterize the regions according to empirical feedback, then sample more heavily from promising regions. To do so, one could probabilistically model the space and sample from the resulting distribution, which would tend to sample in proportion to “goodness” of a region, while efficiently exploring the space.

Because each sample evaluation is costly, one desires to locate regions of high performance with as few queries as possible. Sampling directly from the true distribution would include occasional wasteful samples from low-performance regions, so what we actually desire is a model that not only builds a probabilistic model of the space, but builds a *skewed* model, which disproportionately concentrates mass in promising regions. Sampling from that distribution will be biased towards returning points from high-performance regions. Just as importantly, we desire a method that we can efficiently sample from.

Kernel Density Optimization meets these challenges by building a truncated kernel density estimate model over the hyperparameter cube, where density correlates with empirical performance. The model uses performance-(inversely)proportional bandwidth selection for each observation, tending to concentrate mass in high-performance regions (via small bandwidths) and pushing it away from low-performance regions (via large bandwidths). Furthermore, by truncating the KDE model to the top k performing points, we further bias the model towards concentration of mass to high-performance regions. The end result is a model that iteratively concentrates mass in good regions, is easy to sample from, and naturally balances exploration and exploitation. Figure 10.1 shows the progressive concentration of mass in high-performance regions, reflected in the density of samples returned by the model over a series of five-hundred queries, via snapshots takes during the first fifty queries, the middle fifty, and the final fifty.

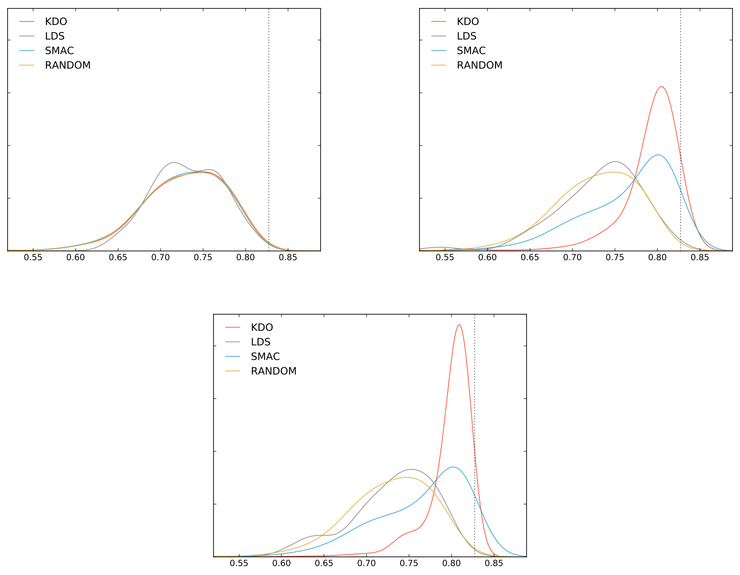


Figure 10.1: Mass concentration around promising regions of the hyperparameter search space. Queries 1-50, 200-250, and 450-500, respectively.

10.2.1 KDO: Mathematical Form

Given an objective function f to be maximized, configurations c_i drawn from the hyperparameter search space, a user-defined truncation length parameter L, a user-defined weight rescaling parameter r, and a user-defined minimum mutation spread parameter m, the truncated weighted KDE that forms the model for KDO sampling of numerical hyperparameters has the following form, assuming configurations $c_{(i)}$ are ranked in descending order of performance:

$$\sum_{i=1}^{L} \frac{w_{(i)}}{\sum_{j=1}^{L} w_{(j)}} K\left(\frac{\|c - c_{(i)}\|}{h_{(i)}}\right) \tag{10.1}$$

where

$$\tau = r \cdot \widehat{F}(f(c_{(1)})), \tag{10.2}$$

$$w_{(i)} = \left(\frac{f(c_{(i)})}{\sum_{k=1}^{L} f(c_{(k)})}\right)^{\tau}, \tag{10.3}$$

$$s_{(i)} = \max(1 - \widehat{F}(f(c_{(i)})), m), \tag{10.4}$$

$$h_{(i)}^2 = (4/(d+2))^{1/(d+4)}\, s_{(i)} n^{-1/(d+4)}. \tag{10.5}$$

Categorical hyperparameters are handled separately, using a set of independent categorical distribution models. For each categorical hyperparameter y with set of options V, the categorical distribution over options v in V is:

$$\Pr(v) = \frac{\alpha + \sum_{i:c_i[y]=v}^{N} f(c_i)}{\alpha|V| + \sum_{v' \in V} \sum_{i:c_i[y]=v'}^{N} f(c_i)}, \tag{10.6}$$

where α is the pseudocount weight, $c_i[y]$ is the value of hyperparameter y for the ith configuration sampled, and N is the total number of configurations sampled.

Pseudocode for KDO is provided in Algorithm 2.

10.3 KDO: Algorithm Details

10.3.1 Initialization

KDO, being a spatial method, begins by converting the hyperparameter search space into a d-dimensional unit cube space, with numerical hyperparameters rescaled to lie between zero and one. (Categorical variables are handled separately from the numerical parameters.) To begin to model the space, a set of N_{init} points is drawn from the cube uniformly at random, with remaining categorical hyperparameters being selected independently from uniform categorical distributions. These N_{init} points are evaluated according to the fitness function (which in most cases is the empirical algorithm performance, such as loss or AUC), and are added to the history. We define this number N_{init} as a user-defined parameter.

Algorithm 2 KDO method for hyperparameter optimization.

```
1: procedure KDO
2: Initialize:
3:     Rescale numeric hyperparameters to [0,1] range.
4:     Randomly sample $N_{\text{init}}$ configurations, save as $pop$.
5:     $history \leftarrow \bigcup_{c \in pop} \langle c, f(c) \rangle$, for metric $f$.
6:     $\widehat{F} \leftarrow \texttt{CDF}(\texttt{mean}(pop), \texttt{var}(pop))$.
7: Main Loop:
8:     for total number of queries do
9:         $kbest \leftarrow \texttt{GetBest}(history, L)$.
10:        for number of samples in batch do
11:            Sample uniformly random, with prob $q$.
12:            Sample from $kbest$, with prob $(1 - q)$.
13:        end for
14:        for each uniformly sampled configuration $c$ do
15:            Update $\widehat{F}$ with $f(c)$.
16:            $history \leftarrow history \bigcup \langle c, f(c) \rangle$.
17:        end for
18:        for each configuration $p$ sampled from $kbest$ do
19:            $c \leftarrow \texttt{mutate}(p)$.
20:            Update $\widehat{F}$ with $f(c)$.
21:            $history \leftarrow history \bigcup \langle c, f(c) \rangle$.
22:        end for
23:    end for
24: Return:
25:    Return $c_{\text{best}} = \texttt{GetBest}(history, 1)$.
26: end procedure
```

10.3.2 Fitness Models

KDO employs two primary models, a simple one-dimensional Gaussian model of fitness values for hyperparameter configurations (based on an estimate constructed from uniformly sampled points), and a weighted kernel density estimate that models the spatial distribution of fitness values within the hyperparameter cube. We will discuss each model in turn.

For the one-dimensional Gaussian model, the initial random configurations and an additional set of interleaved uniform random points (which are taken with probability q, as a user defined parameter, $q = 0.05$ being the default) are used to estimate the mean and variance of this distribution. Evaluating performance values using the CDF of the Gaussian gives a rough estimate of the empirical "goodness" of a configuration, which is then used to control the mutation rate in an inversely fitness-proportional manner. Thus, when a configuration exhibits good performance, the mutation rate is lowered to sample configurations near this point, which fits the assumption that nearby points have similar fitness values. (For minimizing performance metrics such as loss functions, we take advantage of the symmetry of the Gaussian to reflect the point around the mean to obtain the corresponding upper-tail CDF value.)

For the second model, an approximate weighted kernel density estimate over the space of numerical hyperparameters is used, with weights being proportional to the (normalized) empirical performance values, and with fitness-dependent individual covariance matrices at each sampled point. We simplify the model by using a diagonal bandwidth matrix, and set the diagonal entries using Silverman's rule-of-thumb with identical standard deviations set to $s = \max(1 - \widehat{F}(f(c)), m)$ where $\widehat{F}$ is the CDF for the empirical one-dimensional Gaussian fitness model, $f(c)$ is the observed performance of configuration c, and m is the minimum mutation spread, a user-defined parameter in the range $[0, 1]$. Setting the m value lower allows for better exploitation and more precise exploration, at the expense of possible premature convergence. Formally, the bandwidth matrix is defined as a d-dimensional diagonal matrix with diagonal entries equal to $(4/(d+2))^{1/(d+4)}\, sn^{-1/(d+4)}$. This is used as the covariance matrix for each multivariate Gaussian centered at the observed configurations. Thus, the model concentrates mass more closely around strongly performing configurations in the space, and spreads mass away from weakly performing points. Because of the diagonal bandwidth assumption, sampling from the model is also greatly simplified.

An additional set of models are used for categorical hyperparameters. For each categorical hyperparameter, KDO uses a categorical distribution over the categories, where mass is proportional to the normalized sum of historical performance values for each category. (When using minimizing metrics, such as loss, the final normalized weights are inverted by subtracting each from 1 and renormalizing.) Pseudocounts are used for unseen parameter values, with the default being set to 0.1 mass added to each category. Alternative schemes, such as adding $1/n$, may also be used.

10.3.3 Sampling Points

The ease with which one can sample from kernel densities is one of the primary motivations for KDO. To sample from a kernel density estimator, we pick one of the observations uniformly at random, then sample from the Gaussian centered at that point. Furthermore, because of our simplification of the bandwidth matrix, sampling from the multivariate Gaussian becomes equivalent to sampling from independent Gaussians for each numerical hyperparameter.

To select a configuration from the history, we first truncate the history to some predetermined number of empirically strongest samples (controlled by a history length parameter, with default value of $L = 20$), then form a normalized categorical distribution over these configurations, with mass being proportional to the rescaled sum of empirical performance values. Because we want to concentrate mass on strongly performing configurations, we rescale the categorical distribution by raising each entry to a positive power $\tau = r \cdot \widehat{F}(f(c_{(1)}))$, where r is a user-defined weight rescaling parameter (defaulted to $r = 30$), $\widehat{F}$ is again the one-dimensional Gaussian fitness CDF, and $f(c_{(1)})$ is the observed performance of the empirically best configuration in the history (using the reflected quantile when working with minimizing metrics). This rescaling has the effect of concentrating mass more tightly around the best performing configurations, trading exploitation for weaker exploration. It also intensifies the concentration as the quality of the best configuration found so far increases. The weights are then renormalized to create a proper categorical distribution, from which we sample one or more configurations.

For each configuration chosen, we then sample from the multivariate Gaussian centered on that configuration using the simplified bandwidth matrix and sample from the independent categorical distributions for the remaining categorical hyperparameters.

10.3.4 Evaluation and Model Updating

Each configuration sampled from the space is then evaluated according to the performance metric (typically by using the hyperparameters for a machine learning model that is then trained and tested), and the configuration with its accompanying observed performance is added to the history, effectively updating the model. The process then repeats by sampling another set of points from the updated model.

10.4 Extensions

10.4.1 Hierarchical Hyperparameter Spaces

Tree-Structured Parzen Estimators (TPE) [9] and SMAC are both well-suited for optimization of hierarchically structured hyperparameter spaces, allowing for the arbitrary nesting of hyperparameters. Since KDO's model allows for fast sampling of high-dimensional spaces (for example, on a 1000 dimensional Rastrigin function space KDO takes approximately 0.03 seconds to sample and evaluate each configuration), an obvious solution for handling hierarchically structures spaces is simply to optimize all sets of available conditional hyperparameters in parallel, and allow the evaluation algorithm to select out the subset of relevant parameters using the conditional hierarchy. Since each relevant hyperparameter will have a value (as all possible hyperparameters are assigned values), this is guaranteed to return a valid configuration at each iteration. Furthermore, truly irrelevant hyperparameters that are never chosen will not harm predictive performance, and will only negligibly affect computational runtime. Thus, KDO's fast sampling structure allows for natural adaptation to conditional hyperparameter spaces.

10.4.2 Acquisition Functions

Sequential model-based optimization methods often use a secondary acquisition function to estimate the expected improvement of configurations over incumbent configurations (e.g., SMAC and TPE). For SMAC, this involves taking the empirical mean and variance from trees within its random forest to parameterize a Gaussian model for computation of the EI (expected improvement) [39]. TPE takes the results from two configuration models (one for good configurations, one for poor), using a transformed likelihood-ratio of them to estimate the EI [9]. In both cases, proposed configurations are measured according to the acquisition function, effectively filtering which configurations to accept and which to reject. Although the goal for both methods is to maximize the acquisition function, neither does so analytically, instead relying on an ad hoc search procedure to locate configurations with large EI.

Instead of using the indirect process of approximately maximizing an acquisition function, KDO builds a probabilistic model directly on the hyperparameter configuration space to directly sample promising regions. If a secondary acquisition function is desired, one can be accommodated within KDO by using distance-based measures (euclidean distances for numerical hyperparameters, hamming distances for categorical hyperparameters) with empirically estimated Lipschitz smoothness constraints. First, a maximum empirical Lipschitz smoothness constant is estimated using a random subsample of observations. Next, compute the upper Lipschitz bounds for a potential configuration using the k nearest points. Taking the mean and variance of these upper bounds, one can then use a Gaussian model in the manner of [39] to compute an EI score. These scores can then be used to filter out unpromising configurations and select locally maximal ones. Whether using a secondary acquisition function would further improve the performance of KDO is an open question, and more research is needed to determine the appropriateness of such extensions.

10.4.3 Parallelization

One advantage of uniform random sampling over sequential model-based optimization methods is the unquestionably parallel nature of random sampling. Contrast that with the serially processed sequential model updating of SMBO methods. Although sequential, parallelization *can* be introduced into SMBO methods at the expense of sampling from less accurate models. (Indeed, we adopt one such approach for our experiments.) For example, rather than sampling a single configuration from a model at each time step, one can sample several in parallel, as in batch methods. This parallelization comes at a cost: if the samples in a batch were processed serially then latter samples would benefit from information returned from earlier samples. Thus, the trade-off is one of accuracy for speed, with less accurate models being traded for faster evaluations.

A second option for parallelization is to run several independent serial KDO processes, and take the best configuration found from any run. To the degree that one process would not benefit from the information gained by another process, the trade-off between speed and information would still continue to hold. A thorough study of parallelization within KDO remains an open future research area.

10.5 Experiments

10.5.1 Experimental Setup

We evaluated KDO and a number of other hyperparameter optimization methods on a test suite of seven algorithm / dataset pairs. We restricted ourselves to binary classification tasks, and selected three types of learning algorithms to train and test (MART gradient boosted trees [29], logistic regression [3], and averaged perceptrons [28]). We performed 100 independent trials of each task for each hyperparameter optimization method, with each method sequentially sampling 500 points from the hyperparameter search space during every trial.

Datasets

For data, we used publicly available UCI datasets [48] (adult-tiny[2], adult [46], breast cancer [13], ionosphere [74], seismic bumps [75]), a two-class version of the CIFAR-10 dataset [47], and a synthetic fifty-dimensional Rastrigin function [60]. The six real-world datasets were assigned to the learning methods in the following ways: (adult-tiny, averaged perceptron), (breast cancer, averaged perceptron), (adult, logistic regression), (ionosphere, logistic regression), (seismic bumps, gradient boosted trees), and (CIFAR-10 two-class, gradient boosted trees). No learning methods were used for the Rastrigin synthetic function, but the hyperparameter optimization methods attempted to optimize the function directly. Existing train/test splits were used whenever available.

Learning Methods and Hyperparameters

For each different learning method, we searched through a space of hyperparameters, with the number of hyperparameters and their permissible ranges set by the user. Table 10.1 lists the hyperparameters used for each task with their ranges.

Hyperparameter Optimization Methods

We compared KDO against four other hyperparameter optimization methods: uniform random sampling [8], low-discrepancy Sobol sequences [77], Nelder-Mead Optimization [61], and SMAC [39], a state-of-the-art sequential model-based optimization method. Since Nelder-Mead only allows for numerical hyperparameters, it was not tested on tasks containing categorical hyperparameters. In addition, since Nelder-Mead terminates early, to compare over the full range of queries the final sampled point was repeated 500-T times, where T represents the iteration of termination.

The same set of parameter settings were used across all tasks for each method. For SMAC, the settings were taken from the defaults suggested in [39], namely, $numtrees = 10$, $N_{\text{init}} = 50$, $local\_search\_pop = 10$, $\epsilon\_local\_search = 1e - 5$, and $split\_ratio = 0.8$, using an implementation coded by the author following [39]. One area of difference with [39] is that our random

[2]This "tiny" dataset was extracted from the adult training dataset [46], and consisted of 250 training and 250 test examples, drawn at random.

Table 10.1: Datasets, learners, and hyperparameters for each experiment task. (log+ denotes log scaling.)

Dataset	**Learner**	**# Training**	**# Test**	**Hyperparameters with Ranges**

Adult-tiny	Averaged Perceptron	250	250	iterations={[1,1000], log+, stepsize=10}
				learning_rate={[0.01, 1]}

Breast Cancer	Averaged Perceptron	350	349	iterations={[1,1000], log+, stepsize=10}
				learning_rate={[0.01, 1]}
				bins={[2,10], numsteps=5}

Adult	Logistic Regression	32561	16281	L1_penalty={[0,2], numsteps=10}
				L2_penalty={[0,2], numsteps=10}
				optimization_tol={{1e-4 | 1e-5 | 1e-6}}
				memory_size={[5,50], stepsize=15}
				dense_optimizer={{- | +}}

Ionosphere	Logistic Regression	246	105	L1_penalty={[0,2], numsteps=10}
				L2_penalty={[0,2], numsteps=10}
				optimization_tol={{1e-4 | 1e-5 | 1e-6}}
				memory_size={[5,50], stepsize=15}
				dense_optimizer={{- | +}}

Seismic Bumps	Gradient Boosted Trees	1809	775	num_trees={[2,200], numsteps=10, log+}
				num_leaves={[2,100], numsteps=10}
				min_docs_per_leaf={[2,50]}
				learning_rate={[0.01,1]}

CIFAR-10 Binary	Gradient Boosted Trees	50000	10000	num_trees={[2,200], numsteps=10, log+}
				num_leaves={[2,100], numsteps=10}
				min_docs_per_leaf={[2,50]}
				learning_rate={[0.01,1]}
				entropy_coeff={[0,1]}
				split_fraction={[0,1]}
				feature_fraction={[0,1]}
				max_bins_per_feature={[4,1000]}

Rastrigin	N/A (Direct Optimization)	N/A	N/A	dim_i={[-5.12,5.12]} for $i = 1, \ldots, 50$.

forest implementation did not include an n_{min} parameter (which controls how many items must be in a node to split), but had a "min docs in leaves" parameter, which controlled the minimum number of items that could be contained in a leaf (which was set to 2).

For KDO, the settings used were $L = 20$, $m = 0.0001$, $N_{\text{init}} = 50$, $q = 0.05$, $r = 30$. Neither set of parameters was tuned to improve performance on individual tasks, instead being held constant across all tasks. Further investigation is needed to determine the sensitivity of the two methods to changes in their parameter settings.

We used batch processing to speed up the experimental runs, using 50 batches of 10 proposed configurations each iteration (where the 10 configurations are simultaneously drawn from the sample model, evaluated, then are used to update the model in a single large update). This trades off efficiency (drawing multiple points at once), for less frequent updating of the models (which cannot benefit the immediate evaluation feedback provided by the first points in a batch). Because we needed to sample 250,000 configurations for each experiment, with each configuration being used to train and test an independent machine learning model, we choose batch processing as an acceptable compromise to keep overall runtime manageable.

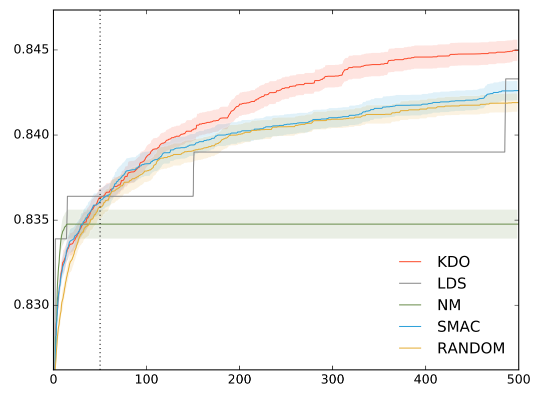
Evaluation Criteria

The area-under-ROC-curve (AUC) was used as the evaluation criterion for all tests. We computed the cumulative maximum AUC over 500 queries for each trial, then found the mean and 95% confidence interval over all trials, plotting the curves visible in Figures 10.2a-10.3.

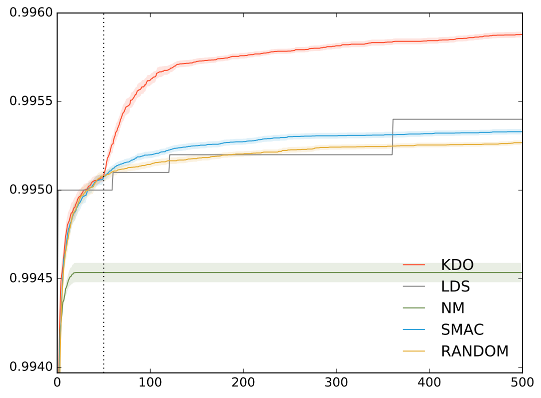
10.5.2 Results

Figure 10.2 plots the outcomes for all six real-world experiments. Some general trends emerge, such as SMAC showing statistically significant improvement over random search and low-discrepancy grid sequences (LDS), as expected, and KDO further improving over SMAC in all real-world tasks. The early improvement of LDS seen in most trials suggests it as a potentially useful method when the total number of iterations is severely restricted, such as when you have a budget of 50 or fewer iterations. The synthetic Rastrigin optimization task was an outlier (Figure 10.3), with low-discrepancy sequences able to achieve the optimum on the first query (being directly in the center of the hypercube, where such sequences coincidentally start), but to make the plots clearer the LDS line is not shown. We also see Nelder-Mead excel on that same function, with SMAC eventually outperforming KDO. In all other experiments, Nelder-Mead performs relatively poorly.

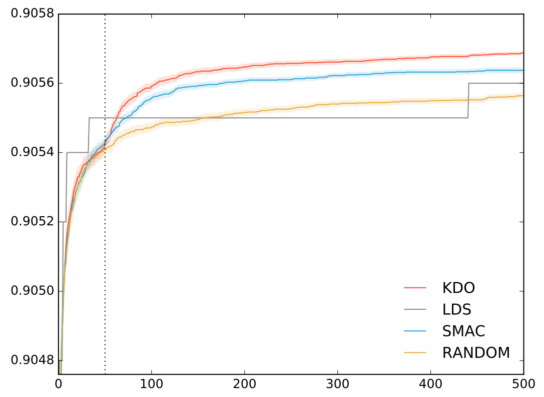
KDO performs significantly better on all real-world experiments, with confidence intervals clearly separated from the nearest competitors (SMAC, LDS, uniform random sampling). Just as SMAC significantly improves over uniform random sampling, KDO further increases that improvement by 64%, on average over all trials and queries.



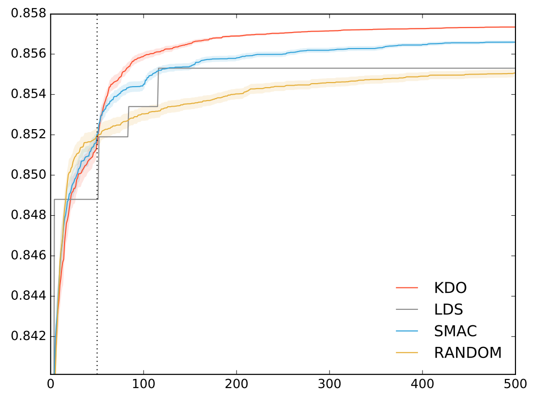
(a) Adult-tiny



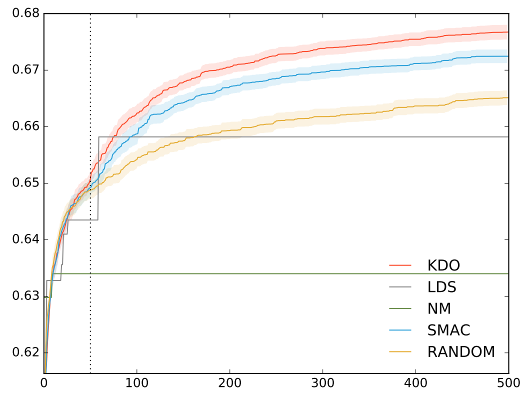
(b) Breast Cancer



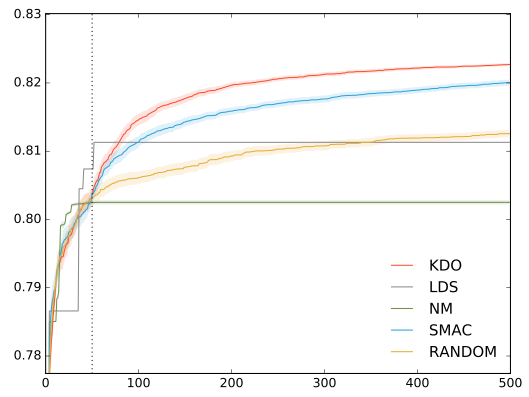
(c) Adult



(d) Ionosphere



(e) Seismic Bumps



(f) CIFAR-10 Binary

Figure 10.2: Results for real-world learning tasks. Dashed lines represent where random initialization ends for SMAC and KDO methods (i.e., 50th query).

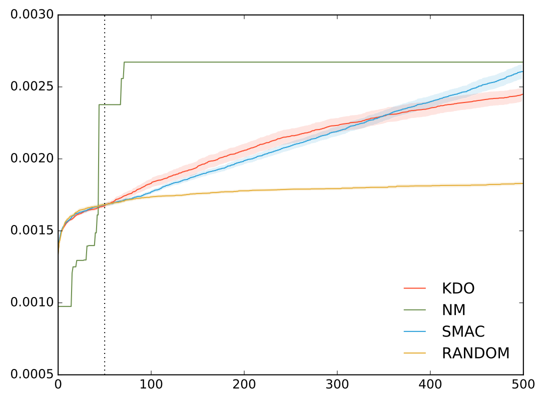


Figure 10.3: Results for synthetic Rastrigin task.

10.6 Discussion

Our theoretical results show the importance of dependence between targets and information resources for successful learning. As discussed in Section 7.6.3, smoothness can be viewed as an exploitable source of dependence, either temporally or spatially. In Chapter 9 we exploited temporal smoothness through the use of temporally regularized methods, allowing our algorithm to perform well in cases where such inductive bias aligned to what held in the real world. Here in Chapter 10, we demonstrate the benefits of exploiting spatial smoothness for improved performance. In both cases the source of improved performance was correctly identified dependence within the respective domains; our view of learning as a search for exploitable dependence works to guide the discovery of novel algorithms by suggesting sources of exploitable information, such as smoothness. While the algorithms presented in these two chapters are not derived directly from the search framework, their discovery was guided by the insight it provides and the structures it suggests.

10.7 Related Work

Several researchers have explored the challenging problem of hyperparameter optimization. Shariari et al. [72] provide a recent overview of Bayesian optimization for automated hyperparameter parameter tuning, covering the history of the problem domain as well as reviewing several relevant approaches. Bergstra and Bengio [8] explored the use of randomized searching within the hyperparameter space, showing its efficiency versus manual and grid-search methods due to the inherent low-effective dimensionality of many hyperparameter response functions of interest. Hutter et al. [39] developed a sequential model-based optimization method SMAC, which allowed them to optimize combinations of categorical and numeric parameters, while improving performance over the state-of-the-art. Bergstra et al. [9] introduced Tree-Structured Parzen Estimators (TPE) as an alternative approach to Gaussian Processes for surrogate function modeling. TPE uses sampling from nonparametric density estimates on the feature space to propose candidate configurations and estimate their expected improvement over incumbent configurations. Bergstra, Yamins and Cox [7] emphasized the importance of hyperparameter choice in algorithm comparisons, arguing that hand-tuned parameters cause poor reproducibility and lead to large variance in performance estimation.

Thorton et al. [80] developed *Auto-WEKA* as an automated sequential model-based Bayesian optimization algorithm for simultaneously selecting algorithms and hyperparameters in WEKA [37]. Their approach used both popular Sequential Model-based Algorithm Configuration (SMAC) and tree-structured Parzen Estimators (TPE), and found they were able to select hyperparamater settings that improved performance over random and grid-based search approaches. Eggensperger et al. [20] introduced a library of benchmarks for evaluating hyperparameter optimization methods and compared SMAC, TPE, and Spearment, three prominent methods for Bayesian hyperparameter optimization. Feurer et al. [25] addressed the problem of "cold-starts" in Bayesian hyperparameter optimization, suggesting a form of transfer learning to initialize the SMAC procedure, showing improved rates of convergence towards sets of optimal parameters.

Snoek et al. [76] developed practical Bayesian methods for hyperparameter optimization based on a Gaussian processes model with a modified acquisition function, that optimized expected improvement per second (of wall-clock time). Wang et al. [86] proposed using random embeddings for Bayesian hyperparameter optimization in high dimensions, demonstrating the effectiveness of this technique (better than random search and on par with SMAC) on synthetic and real-world datasets. Hutter, Hoos and Leyton-Brown [40] introduced an efficient approach for assessing the importance of individual (as well as subsets of) hyperparameters using a form of functional ANOVA on estimated marginals computed from random forest models.

Part IV

Conclusion

Chapter 11

Conclusion

THIS thesis presents a unified search framework for learning in finite spaces, capable of addressing what makes machine learning work while also answering questions related to search and optimization. This is an improvement over previous work as it allows us to address both areas within the same framework instead of having to deal with them separately.

We reformulate many areas of machine learning as searches within our framework, including regression, classification, parameter estimation, clustering, and empirical risk minimization. Exploring two dominant paradigms for understanding machine learning (statistical learning theory and minimum description length) we see that the former is naturally viewed as a search to find models with low risk, while latter can be seen as a specific form of inductive bias. These paradigms thus fit naturally within our higher-level framework, anchoring them to a more general structure. Examining the link between compressibility and generalization, we see that Occam's razor (i.e., biasing models towards simplicity) is not the general explanation for why machine learning works, but at best can be a sufficient condition for learning in certain restricted contexts. Bounds proving good generalization performance for compressed structures are shown to not depend on any Occam-like notion of simplicity, but derive from small set sizes, multiple-hypotheses testing controls, and i.i.d. data assumptions.

Several formal results are proven within this framework. We formally bound the proportion of favorable search problems for any fixed search algorithm, showing that algorithms can only outperform uniform sampling on a limited proportion of problems. Furthermore, the information needed to identify a problem giving b bits of competitive advantage over uniform sampling requires at least b bits – thus, information is conserved. We prove within our framework an early result from Culberson [15] bounding the proportion of problems for which an algorithm is expected to find a target within a fixed number of queries. Defining a *search strategy* as the probability distribution induced by an algorithm over the search space when considering the expected per-query probability of success, we find that for any fixed search problem the proportion of favorable search strategies is also bounded. It follows that whether one fixes the algorithm and varies the search problem or fixes the search problem and varies the algorithm, finding a favorable match is difficult. In the same vein, it is shown in Chapter 7 that any deterministic binary classification algorithm with restricted data set size can only learn a limited number of concepts to low error.

Of central importance, two results are proven that quantify the effects of dependence for learning. In the first, a simple upper bound is given on the expected probability of success for algorithms defined in terms of dependence, target set size, and target predictability. It shows dependence to be a necessary condition for the successful learning of uncertain targets. A closed-form exact expression is also derived for the same quantity, demonstrating that the probability of success is determined by the dependence between information resources (like datasets) and targets, algorithm information loss, the size of targets, target uncertainty, randomness, and target structure. This equation proves the importance of dependence between what is observed and what is latent for better-than-chance learning, and gives an exact formula for computing the expected probability of success for any search algorithm.

Lastly, to demonstrate the practical utility of our paradigm, we apply our insights to two learning areas, time-series segmentation and hyperparameter optimization. Chapters 9 and 10 show how a "dependence-first" view of learning leads naturally to discovering new exploitable sources of dependence, and we operationalize our understanding by developing new algorithms for these application areas with strong empirical performance.

Part V

Appendices

Appendix A

The Need for Biased Classifiers: Noisy Labels

We begin by defining *ψ-expansions*, *ψ-consistency* and *ψ-consistent sets*, which allow us to handle training data with noisy labels. We overload the function ψ as is done in computer programming, where the argument determines which form of the function we are using.

Definition A.0.1. (ψ-expansion of D) Given spaces $\mathcal{X}$ and $\mathcal{Y}$, a set-valued function $\psi : \mathcal{X} \to 2^{\mathcal{Y}}$, and a set D of (x, y) pairs with $x \in \mathcal{X}$ and $y \in \mathcal{Y}$, define the **ψ-expansion of** D as the set

$$\psi(D) = \{(x, y') : (x, y) \in D, y' \in \psi(x) \text{ where } \{y\} \subseteq \psi(x) \subseteq \mathcal{Y}\}.$$

Thus, a ψ-expansion adds additional tuples to the dataset, which represent possible alternative y values for each x, defined by the set-valued function ψ applied to elements of $\mathcal{X}$.

Definition A.0.2. (ψ-consistency with D) Given spaces $\mathcal{X}$ and $\mathcal{Y}$, a set D and its ψ-expansion $\psi(D)$, a function $g : \mathcal{X} \to \mathcal{Y}$ is said to be **ψ-consistent with** D iff $(x, g(x)) \in \psi(D)$ for every $x \in D_x$.

Definition A.0.3. (ψ-consistent set $\Omega_{\psi(D)}$) A **ψ-consistent set** $\Omega_{\psi(D)}$ is defined as the set of all functions $g : \mathcal{X} \to \mathcal{Y}$ that are ψ-consistent with D.

The mapping ψ acts as an expansion, turning a single element y into a set containing y (and possibly other elements of $\mathcal{Y}$). This allows us to consider consistency with training data when the true y value may differ from the observed y value, as when label noise is present. Of course, the noiseless case is also covered in this definition, as a particular special case: the set of hypothesis functions consistent with training set D, denoted Ω_D, is equivalent to the ψ-consistent set $\Omega_{\psi(D)}$ when $\psi(x) = \{y\}$ for every (x, y) pair in D (or equivalently, when $\psi(D) = D$).

Theorem 14. *Define as follows:*

- $\mathcal{X}$ *- finite instance space,*
- $\mathcal{Y}$ *- finite label space,*
- Ω *-* $\mathcal{Y}^{\mathcal{X}}$*, the space of possible concepts on* $\mathcal{X}$*,*
- h *- a hypothesis,* $h \in \Omega$*,*
- $D = \{(x_1, y_1), \ldots, (x_N, y_N)\}$ *- any training dataset where* $x_i \in \mathcal{X}$*,* $y_i \in \mathcal{Y}$*,*
- $D_x = \{x : (x, \cdot) \in D\}$ *- the set of* x *instances in* D*,*

- $V_x = \{x_1, \ldots, x_M\}$ - *any test dataset disjoint from* D_x *(i.e.,* $D_x \cap V_x = \emptyset$*) containing exactly* M *elements* $x_i \in \mathcal{X}$ *with* $M > 0$*, hidden from the algorithm during learning,*
- $\Omega_{\psi(D)}$ - *subset of hypotheses* ψ*-consistent with* D *for some* $\psi(\cdot)$ *function, and*
- unbiased classifier $\mathcal{A}$ - *any classifier such that* $P(h|D,M) = \mathbb{1}(h \in \Omega_{\psi(D)})/|\Omega_{\psi(D)}|$ *(i.e., makes no assumptions beyond strict* ψ*-consistency with training data).*

Then the distribution of 0-1 generalization error counts for any unbiased classifier is given by

$$P(w|D,M;\mathcal{A}) = \binom{M}{w}\left(1 - \frac{1}{|\mathcal{Y}|}\right)^{w}\left(\frac{1}{|\mathcal{Y}|}\right)^{M-w}$$

where w *is the number of wrong predictions on disjoint test sets of size* M*. Thus, every unbiased classifier has generalization performance equivalent to random guessing (e.g., flipping a coin) of class labels for unseen instances.*

Proof. In agreement with the problem model given in Figure 5.1, we assume the training data D are generated from the true concept h^* by some process, and that h is chosen by $\mathcal{A}$ using D alone, without access to h^*. Thus, $h^* \to D \to h$, implying $P(h^*|D,h,M) = P(h^*|D,M)$ by d-separation. By similar reasoning, since $\{D_x, M\} \to V_x$, with D as an ancestor of both V_x and h and no other active path between them, we have $P(V_x|f,D,h,M) = P(V_x|f,D,M)$. This can be verified intuitively by the fact that V_x is generated prior to the algorithm choosing h, and the algorithm has no access to V_x when choosing h (it only has access to D, which we're already conditioning on).

Let $K = \mathbb{1}(h \in \Omega_{\psi(D)})/|\Omega_{\psi(D)}|$ and let L be the number of free instances in $\mathcal{X}$, namely

$$L = |\mathcal{X}| - (|D_x| + |V_x|) \tag{A.1}$$

$$= |\mathcal{X}| - (N + M). \tag{A.2}$$

Then

$$P(w|D,M;\mathcal{A}) = \frac{P(w,D|M)}{P(D|M)} \tag{A.3}$$

$$= \frac{\sum_{h\in\Omega_{\psi(D)}} P(w,h,D|M)}{\sum_{h\in\Omega_{\psi(D)}} P(h,D|M)} \tag{A.4}$$

$$= \frac{\sum_{h\in\Omega_{\psi(D)}} P(w|h,D,M)P(h|D,M)P(D|M)}{\sum_{h\in\Omega_{\psi(D)}} P(h|D,M)P(D|M)} \tag{A.5}$$

$$= \frac{KP(D|M)\sum_{h\in\Omega_{\psi(D)}} P(w|h,D,M)}{KP(D|M)|\Omega_{\psi(D)}|} \tag{A.6}$$

$$= \frac{1}{|\Omega_{\psi(D)}|}\sum_{h\in\Omega_{\psi(D)}} P(w|h,D,M) \tag{A.7}$$

$$= \frac{1}{|\mathcal{Y}|^{M+L}\prod_{i=1}^{N}|\psi(x_i)|}\sum_{h\in\Omega_{\psi(D)}} P(w|h,D,M) \tag{A.8}$$

$$= \frac{1}{C_1|\mathcal{Y}|^{M+L}}\sum_{h\in\Omega_{\psi(D)}} P(w|h,D,M), \tag{A.9}$$

where we have defined $C_1 := \prod_{i=1}^{N} |\psi(x_i)|$. Marginalizing over possible true concepts f for term $P(w|h, D, M)$ and letting $Z = \{f, h, D, M\}$, we have

$$P(w|h, D, M) = \sum_{f \in \Omega} P(w, f|h, D, M) \tag{A.10}$$
$$= \sum_{f \in \Omega} P(w|Z)P(f|h, D, M) \tag{A.11}$$
$$= \sum_{f \in \Omega} P(w|Z)P(f|D, M) \quad \text{(by d-separation)} \tag{A.12}$$
$$= \sum_{f \in \Omega} P(f|D, M) \sum_{v_x} P(w, v_x|Z) \tag{A.13}$$
$$= \sum_{f \in \Omega} P(f|D, M) \sum_{v_x} P(v_x|Z)P(w|Z, v_x) \tag{A.14}$$
$$= \sum_{f \in \Omega} P(f|D, M) \sum_{v_x} P(v_x|Z)\mathbb{1}(w = w_{h,f}(v_x)), \tag{A.15}$$

where $w_{h,f}(v_x) = \sum_{x \in v_x} \mathbb{1}(h(x) \neq f(x))$ and the final equality follows since

$$P(w|Z, v_x) = P(w|f, h, D, M, v_x) = \begin{cases} 1 & w = w_{h,f}(v_x), \\ 0 & w \neq w_{h,f}(v_x). \end{cases} \tag{A.16}$$

Combining (A.9) and (A.15), we obtain

$$P(w|D, M; \mathcal{A}) = \frac{1}{C_1 |\mathcal{Y}|^{M+L}} \sum_{h \in \Omega_{\psi(D)}} \left[\sum_{f \in \Omega} P(f|D, M) \sum_{v_x} P(v_x|f, h, D, M)\mathbb{1}(w = w_{h,f}(v_x)) \right] \tag{A.17}$$
$$= \frac{1}{C_1 |\mathcal{Y}|^{M+L}} \sum_{h \in \Omega_{\psi(D)}} \left[\sum_{f \in \Omega} P(f|D, M) \sum_{v_x} P(v_x|f, D, M)\mathbb{1}(w = w_{h,f}(v_x)) \right] \tag{A.18}$$
$$= \frac{1}{C_1 |\mathcal{Y}|^{M+L}} \sum_{f \in \Omega} P(f|D, M) \sum_{v_x} P(v_x|Z') \sum_{h \in \Omega_{\psi(D)}} \mathbb{1}(w = w_{h,f}(v_x)), \tag{A.19}$$

where we have defined $Z' := \{f, D, M\}$ and the second equality follows from d-separation between V_x and h, conditioned on D.

Note that $\sum_{h \in \Omega_{\psi(D)}} \mathbb{1}(w = w_{h,f}(v_x))$ is the number of hypotheses ψ-consistent with D that disagree with concept f exactly w times on v_x. There are $\binom{M}{w}$ ways to choose w disagreements with f on v_x, and for each of the w disagreements we can choose $|\mathcal{Y}| - 1$ possible values for h at that instance, giving a multiplicative factor of $(|\mathcal{Y}| - 1)^w$. For the training set D, there are exactly $|\psi(x_i)|$ alternative values for each training set instance x_i, giving a multiplicative factor of $C_1 = \prod_{i=1}^{N} |\psi(x_i)|$. For the remaining L instances that are in neither D nor v_x, we have $|\mathcal{Y}|$

possible values, giving the additional multiplicative factor of $|\mathcal{Y}|^L$. Thus,

$$P(w|D,M;\mathcal{A}) = \frac{1}{C_1|\mathcal{Y}|^{M+L}} \sum_{f\in\Omega} P(f|D,M) \sum_{v_x} P(v_x|Z') \left[\binom{M}{w} C_1(|\mathcal{Y}|-1)^w |\mathcal{Y}|^L\right] \tag{A.20}$$

$$= \frac{1}{C_1|\mathcal{Y}|^{M+L}} \left[\binom{M}{w} C_1(|\mathcal{Y}|-1)^w |\mathcal{Y}|^L\right] \sum_{f\in\Omega} P(f|D,M) \sum_{v_x} P(v_x|Z') \tag{A.21}$$

$$= \frac{1}{C_1|\mathcal{Y}|^{M}|\mathcal{Y}|^{L}} \left[\binom{M}{w} C_1(|\mathcal{Y}|-1)^w |\mathcal{Y}|^L\right] \sum_{f\in\Omega} P(f|D,M) \tag{A.22}$$

$$= \binom{M}{w}(|\mathcal{Y}|-1)^w \frac{1}{|\mathcal{Y}|^M} \tag{A.23}$$

$$= \binom{M}{w}\left[|\mathcal{Y}|\left(1-\frac{1}{|\mathcal{Y}|}\right)\right]^w \frac{1}{|\mathcal{Y}|^M} \tag{A.24}$$

$$= \binom{M}{w}\left(1-\frac{1}{|\mathcal{Y}|}\right)^w \frac{|\mathcal{Y}|^w}{|\mathcal{Y}|^M} \tag{A.25}$$

$$= \binom{M}{w}\left(1-\frac{1}{|\mathcal{Y}|}\right)^w |\mathcal{Y}|^{w-M} \tag{A.26}$$

$$= \binom{M}{w}\left(1-\frac{1}{|\mathcal{Y}|}\right)^w \left(\frac{1}{|\mathcal{Y}|}\right)^{M-w}. \tag{A.27}$$

□

Bibliography

[1] Lee Altenberg. The schema theorem and prices theorem. *Foundations of genetic algorithms*, 3:23–49, 1995. 2.3.3

[2] Kerem Altun, Billur Barshan, and Orkun Tunçel. Comparative study on classifying human activities with miniature inertial and magnetic sensors. *Pattern Recogn.*, 43(10):3605–3620, October 2010. ISSN 0031-3203. doi: 10.1016/j.patcog.2010.04.019. URL `http://dx.doi.org/10.1016/j.patcog.2010.04.019`. 9.4.1

[3] Galen Andrew and Jianfeng Gao. Scalable training of l 1-regularized log-linear models. In *Proceedings of the 24th international conference on Machine learning*, pages 33–40. ACM, 2007. 10.5.1

[4] David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007. 4.3

[5] Anne Auger and Olivier Teytaud. Continuous lunches are free! In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 916–922. ACM, 2007. 2.3.8, 2.3.9

[6] Toby Berger. Rate-distortion theory. *Encyclopedia of Telecommunications*, 1971. 7.6.3

[7] J Bergstra, D Yamins, and DD Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *Proc. 30th International Conference on Machine Learning (ICML-13)*, 2013. 10, 10.7

[8] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *The Journal of Machine Learning Research*, 13(1):281–305, 2012. 10, 10.2, 10.5.1, 10.7

[9] James S Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems*, pages 2546–2554, 2011. 10, 10.4.1, 10.4.2, 10.7

[10] Christopher M Bishop. *Pattern Recognition and Machine Learning*. Springer, 2007. 616–625. 9.3

[11] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K Warmuth. Occam's razor. *Information processing letters*, 24(6):377–380, 1987. 7.5

[12] Antoine Bordes, Seyda Ertekin, Jason Weston, and Léon Bottou. Fast kernel classifiers with online and active learning. *Journal of Machine Learning Research*, 6(Sep):1579–1619, 2005. 7.2

[13] Peter Clark and Tim Niblett. Induction in noisy domains. In *Progress in Machine Learning (from the Proceedings of the 2nd European Working Session on Learning)*, volume 96, pages 11–30. Sigma Press, 1987. 10.5.1

[14] D. Cox and N. Pinto. Beyond simple features: A large-scale feature search approach to unconstrained face recognition. In *Automatic Face Gesture Recognition and Workshops (FG 2011), 2011 IEEE International Conference on*, pages 8–15, March 2011. doi: 10.1109/FG.2011.5771385. 10

[15] J.C. Culberson. On the futility of blind search: An algorithmic view of 'no free lunch'. *Evolutionary Computation*, 6(2):109–127, 1998. 1.1, 2.3.3, 5.4, 11

[16] Rachel Cummings, Katrina Ligett, Kobbi Nissim, Aaron Roth, and Zhiwei Steven Wu. Adaptive learning with robust generalization guarantees. In *Proceedings of the 29th Conference on Learning Theory, COLT*, pages 23–26, 2016. 7.2

[17] W.A. Dembski and R.J. Marks II. Conservation of information in search: Measuring the cost of success. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 39(5):1051 –1061, sept. 2009. ISSN 1083-4427. doi: 10.1109/TSMCA.2009.2025027. 1.1, 3.3, 5.1, 5.2, 2

[18] William A. Dembski, Winston Ewert, and Robert J. Marks II. A general theory of information cost incurred by successful search. In *Biological Information*, chapter 3, pages 26–63. World Scientific, 2013. doi: 10.1142/9789814508728_0002. 5.1

[19] Pedro Domingos. The role of occam's razor in knowledge discovery. *Data mining and knowledge discovery*, 3(4):409–425, 1999. 7.2, 7.5, 7.6

[20] K. Eggensperger, M. Feurer, F. Hutter, J. Bergstra, J. Snoek, H. Hoos, and K. Leyton-Brown. Towards an empirical foundation for assessing bayesian optimization of hyperparameters. In *NIPS workshop on Bayesian Optimization in Theory and Practice*, 2013. 10.7

[21] T.M. English. Evaluation of evolutionary and genetic optimizers: No free lunch. In *Evolutionary Programming V: Proceedings of the Fifth Annual Conference on Evolutionary Programming*, pages 163–169, 1996. 5.1

[22] T.M. English. No more lunch: Analysis of sequential search. In *Evolutionary Computation, 2004. CEC2004. Congress on*, volume 1, pages 227–234. IEEE, 2004. 1, 1.1

[23] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD*, volume 96, pages 226–231, 1996. 4.3

[24] Robert M Fano and David Hawkins. Transmission of information: A statistical theory of communications. *American Journal of Physics*, 29(11):793–794, 1961. 5.6

[25] M. Feurer, T. Springenberg, and F. Hutter. Initializing bayesian hyperparameter optimization via meta-learning. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, January 2015. 10, 10.7

[26] Emily B. Fox and Erik B. Sudderth. HDP-HMM Toolbox. `https://www.stat.washington.edu/~ebfox/software.html`, 2009. URL `https://`

www.stat.washington.edu/~ebfox/software.html. [Online; accessed 20-July-2014]. 9.4.2

[27] Emily B Fox, Erik B Sudderth, Michael I Jordan, Alan S Willsky, et al. A sticky HDP-HMM with application to speaker diarization. *The Annals of Applied Statistics*, 5(2A): 1020–1056, 2011. 9, 9.4, 9.6

[28] Yoav Freund and Robert E Schapire. Large margin classification using the perceptron algorithm. *Machine learning*, 37(3):277–296, 1999. 10.5.1

[29] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001. 10.5.1

[30] Jean-luc Gauvain and Chin-hui Lee. Maximum A Posteriori Estimation for Multivariate Gaussian Mixture Observations of Markov Chains. *IEEE Transactions on Speech and Audio Processing*, 2:291–298, 1994. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.18.6428. 9.2, 9.6

[31] Corrado Gini. On the measure of concentration with special reference to income and statistics. In *Colorado College Publication*, number 208 in General Series, pages 73–79, 1936. 9.3.1

[32] Peter Grunwald. A tutorial introduction to the minimum description length principle. *arXiv preprint math/0406077*, 2004. (document), 7.6, 7.2, 7.6.1, 7.6.2

[33] Peter Grünwald and John Langford. Suboptimal behavior of bayes and mdl in classification under misspecification. *Machine Learning*, 66(2-3):119–149, 2007. 7.6.2

[34] Peter D Grünwald. *The minimum description length principle*. MIT press, 2007. 2.1, 7.6

[35] I. Guyon, K. Bennett, G. Cawley, H. J. Escalante, S. Escalera, Tin Kam Ho, N. Maci, B. Ray, M. Saeed, A. Statnikov, and E. Viegas. Design of the 2015 chalearn automl challenge. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, July 2015. doi: 10.1109/IJCNN.2015.7280767. 5.5

[36] John H Holland. Adaptation in natural and artificial systems. an introductory analysis with application to biology, control, and artificial intelligence. *Ann Arbor, MI: University of Michigan Press*, 1975. 2.3.3

[37] Geoffrey Holmes, Andrew Donkin, and Ian H Witten. Weka: A machine learning workbench. In *Intelligent Information Systems, 1994. Proceedings of the 1994 Second Australian and New Zealand Conference on*, pages 357–361. IEEE, 1994. 10.7

[38] David Hume. A Treatise of Human Nature by David Hume, reprinted from the original edition in three volumes and edited, with an analytical index, 1896. 2.1

[39] Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration (extended version). Technical report, Technical Report TR-2010-10, University of British Columbia, Computer Science, 2010. 10, 10.4.2, 10.5.1, 10.7

[40] Frank Hutter, Holger Hoos, and Kevin Leyton-Brown. An efficient approach for assessing hyperparameter importance. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 754–762, 2014. 10.7

[41] Marcus Hutter. Optimality of universal bayesian sequence prediction for general loss and alphabet. *Journal of Machine Learning Research*, 4(Nov):971–1000, 2003. 2.1

[42] Christian Igel and Marc Toussaint. On classes of functions for which no free lunch results hold. *arXiv preprint cs/0108011*, 2001. 2.3.5

[43] Edwin T Jaynes. *Probability theory: The logic of science*. Cambridge university press, 2003. 7.6

[44] Mark Johnson. Why doesn't EM find good HMM POS-taggers. In *In EMNLP*, pages 296–305, 2007. 9.6

[45] Yoshinobu Kawahara, Takehisa Yairi, and Kazuo Machida. Change-point detection in time-series data based on subspace identification. In *Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on*, pages 559–564. IEEE, 2007. 9

[46] 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*, volume 96, pages 202–207. Citeseer, 1996. 10.5.1, 2

[47] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009. 10.5.1

[48] M. Lichman. UCI machine learning repository, 2013. URL `http://archive.ics.uci.edu/ml`. 10.5.1

[49] Nick Littlestone and Manfred Warmuth. Relating data compression and learnability. Technical report, Technical report, University of California, Santa Cruz, 1986. 7.2, 8, 7.2

[50] Song Liu, Makoto Yamada, Nigel Collier, and Masashi Sugiyama. Change-point detection in time-series data by relative density-ratio estimation. *Neural Networks*, 43:72–83, 2013. 9

[51] Alan J Lockett and Risto Miikkulainen. A probabilistic re-formulation of no free lunch: Continuous lunches are not free. *Evolutionary Computation*, 2016. 2.3.8, 2.3.9

[52] James AR Marshall and Thomas G Hinton. Beyond no free lunch: realistic algorithms for arbitrary problem classes. In *IEEE Congress on Evolutionary Computation*, pages 1–6. IEEE, 2010. 2.3.7

[53] Marina Meilă. Comparing clusterings by the variation of information. In Bernhard Schölkopf and ManfredK. Warmuth, editors, *Learning Theory and Kernel Machines*, volume 2777 of *Lecture Notes in Computer Science*, pages 173–187. Springer Berlin Heidelberg, 2003. ISBN 978-3-540-40720-1. doi: 10.1007/978-3-540-45167-9_14. URL `http://dx.doi.org/10.1007/978-3-540-45167-9_14`. 9.4.2

[54] T.M. Mitchell. *Machine Learning*. McGraw-Hill International Editions. McGraw-Hill, 1997. ISBN 9780071154673. URL `https://books.google.com/books?id=EoYBngEACAAJ`. 7.5

[55] Tom M. Mitchell. The need for biases in learning generalizations. Technical report, Rutgers University, 1980. 1, 1.1, 2.4.1, 5.7, 5.8, 10.1

[56] Tom M. Mitchell. Generalization as search. *Artificial intelligence*, 18(2):203–226, 1982.

1, 2.2

[57] George D. Montañez. Bounding the number of favorable functions in stochastic search. In *Evolutionary Computation (CEC), 2013 IEEE Congress on*, pages 3019–3026, June 2013. doi: 10.1109/CEC.2013.6557937. 1.1, 5.2

[58] George D Montañez, Saeed Amizadeh, and Nikolay Laptev. Inertial hidden markov models: Modeling change in multivariate time series. In *Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI 2015)*, pages 1819–1825, 2015. 9

[59] Shay Moran and Amir Yehudayoff. Sample compression schemes for vc classes. *Journal of the ACM (JACM)*, 63(3):21, 2016. 7.2, 7.2

[60] Heinz Mühlenbein, M Schomisch, and Joachim Born. The parallel genetic algorithm as function optimizer. *Parallel computing*, 17(6-7):619–632, 1991. 10.5.1

[61] John A Nelder and Roger Mead. A simplex method for function minimization. *The computer journal*, 7(4):308–313, 1965. 10.5.1

[62] Christoph Neukirchen and Gerhard Rigoll. Controlling the complexity of HMM systems by regularization. *Advances in Neural Information Processing Systems*, pages 737–743, 1999. 9.6

[63] Lawrence Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989. 9.1, 9.6

[64] R Bharat Rao, Diana Gordon, and William Spears. For every generalization action, is there really an equal and opposite reaction? analysis of the conservation law for generalization performance. *Urbana*, 51:61801. 1.1

[65] Bonnie K Ray and Ruey S Tsay. Bayesian methods for change-point detection in long-range dependent processes. *Journal of Time Series Analysis*, 23(6):687–705, 2002. 9

[66] BD Ripley. Neural networks and pattern recognition. *Cambridge University*, 1996. 7.6

[67] Jorma Rissanen. *Stochastic complexity in statistical inquiry*, volume 15. World scientific, 1998. 7.6

[68] N. Sauer. On the density of families of sets. *Journal of Combinatorial Theory, Series A*, 13 (1):145–147, 1972. 5.1, 5.1

[69] C. Schaffer. A conservation law for generalization performance. In W. W. Cohen and H. Hirsch, editors, *Proceedings of the Eleventh International Machine Learning Conference*, pages 259–265. Rutgers University, New Brunswick, NJ, 1994. 1, 2.3.2

[70] Cullen Schaffer. Overfitting avoidance as bias. *Machine learning*, 10(2):153–178, 1993. 2.4.2, 2.4.3, 7.2, 7.6.3

[71] C. Schumacher, MD Vose, and LD Whitley. The no free lunch and problem description length. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 565–570, 2001. 1, 2.3.4, 2.3.6, 5.1

[72] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando de Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2016. 10.7

[73] Cosma Rohilla Shalizi. "occam"-style bounds for long programs, 2016. URL `http://bactra.org/notebooks/occam-bounds-for-long-programs.html`. [Online; accessed 17-March-2017]. 7.5

[74] Vincent G Sigillito, Simon P Wing, Larrie V Hutton, and Kile B Baker. Classification of radar returns from the ionosphere using neural networks. *Johns Hopkins APL Technical Digest*, 10(3):262–266, 1989. 10.5.1

[75] Marek Sikora and Łukasz Wróbel. Application of rule induction algorithms for analysis of data collected by seismic hazard monitoring systems in coal mines. *Archives of Mining Sciences*, 55(1):91–114, 2010. 10.5.1

[76] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pages 2951–2959, 2012. 10, 10.7

[77] Il'ya Meerovich Sobol'. On the distribution of points in a cube and the approximate evaluation of integrals. *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki*, 7(4):784–802, 1967. 10.5.1

[78] Ray Solomonoff. Does algorithmic probability solve the problem of induction. *Information, Statistics and Induction in Science*, pages 7–8, 1996. 2.1

[79] Ray J. Solomonoff. Three kinds of probabilistic induction: Universal distributions and convergence theorems. *The Computer Journal*, 51(5):566–570, 2008. doi: 10.1093/comjnl/bxm120. URL `http://comjnl.oxfordjournals.org/content/51/5/566.abstract`. 2.1

[80] Chris Thornton, Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Auto-weka: Combined selection and hyperparameter optimization of classification algorithms. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 847–855. ACM, 2013. 5.5, 10, 10.7

[81] Vladimir N Vapnik. An overview of statistical learning theory. *IEEE transactions on neural networks*, 10(5):988–999, 1999. 2.1, 4, 4.6, 6, 6.1.1

[82] Vladimir N Vapnik and A Ya Chervonenkis. Necessary and sufficient conditions for the uniform convergence of means to their expectations. *Theory of Probability & Its Applications*, 26(3):532–553, 1982. 6.1

[83] Vladimir N Vapnik and A Ya Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. In *Measures of Complexity*, pages 11–30. Springer, 2015. 6.1

[84] Vladimir Naumovich Vapnik and Vlamimir Vapnik. *Statistical learning theory*, volume 1. Wiley New York, 1998. 7.6

[85] John Vickers. The problem of induction. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Spring 2016 edition, 2016. 2.1

[86] Ziyu Wang, Masrour Zoghi, Frank Hutter, David Matheson, and Nando de Freitas. Bayesian optimization in a billion dimensions via random embeddings. *arXiv preprint arXiv:1301.1942*, 2013. 10, 10.7

[87] Geoffrey I Webb. Further experimental evidence against the utility of occam's razor. *Journal of Artificial Intelligence Research*, 4:397–417, 1996. 7.2, 7.6

[88] Wikipedia. Gini coefficient — Wikipedia, the free encyclopedia, 2014. URL `http://en.wikipedia.org/wiki/Gini_coefficient`. [Online; accessed 8-June-2014]. 9.3.1

[89] Alan S Willsky, Erik B Sudderth, Michael I Jordan, and Emily B Fox. Nonparametric Bayesian learning of switching linear dynamical systems. In *Advances in Neural Information Processing Systems*, pages 457–464, 2009. 9

[90] D Randall Wilson and Tony R Martinez. Bias and the probability of generalization. In *Intelligent Information Systems, 1997. IIS'97. Proceedings*, pages 108–114. IEEE, 1997. 2.4.4

[91] David H Wolpert. On the connection between in-sample testing and generalization error. *Complex Systems*, 6(1):47, 1992. 2.3.1, 5.8.3

[92] David H Wolpert. What the no free lunch theorems really mean; how to improve search algorithms. In *Santa fe Institute Working Paper*, page 12. 2012. 2.3.1

[93] David H Wolpert et al. On overfitting avoidance as bias. Technical report, Technical Report SFI TR 92-03-5001. Santa Fe, NM: The Santa Fe Institute, 1993. 2.4.3, 7.2

[94] D.H. Wolpert. The supervised learning no-free-lunch theorems. In *Proceedings of the 6th Online World Conference on Soft Computing in Industrial Applications*, volume 6, pages 1–20, 2001. 1, 2.3.1, 2.3.2

[95] D.H. Wolpert and W.G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, April 1997. doi: 10.1109/4235.585893. 1, 2.3.1, 5.1

[96] Yao Xie, Jiaji Huang, and Rebecca Willett. Change-point detection for high-dimensional time series with missing data. *Selected Topics in Signal Processing, IEEE Journal of*, 7(1): 12–27, 2013. 9

[97] Tong Zhang. On the convergence of mdl density estimation. In *COLT*, volume 3120, pages 315–330. Springer, 2004. 7.6.1