Senior Thesis in Mathematics

The Limits of Transfer Learning

Author:
Huey Sun

Advisor:
Dr. Ghassan Sarkis

Submitted to Pomona College in Partial Fulfillment
of the Degree of Bachelor of Arts

April 29, 2020

Abstract

This thesis is about transfer learning, the Search Framework, and various information-theoretic bounds to algorithmic success. By defining decomposability, a property of probability-of-success metrics, we can generalize a number of existing results that relate algorithmic performance to features of the Search Framework. By defining affinity, we can discuss the success of transfer learning with respect to the distribution of learned information resources generated by the source problem. Within the framework, I present an upper bound for the probability of success of transfer learning based on the mutual information between the source and recipient problems, and discuss an experimental similarity heuristic that relates a model's testing accuracy on the recipient problem to the efficacy of transfer.

Contents

Foreword

There seems to be an immense paranoia in the cultural consciousness when it comes to artificial intelligence. It's almost commonly accepted by the casual observer that the development of general AI, machines with humanesque levels of understanding and capability, is just a matter of time — Elon Musk declared that robots will supersede humans by the end of our lifetimes[1], articles are devoted to fear-mongering the day your smart toaster goes rogue, and visions of the future are rife with predictions of some kind of Skynet-related apocalypse.

And there certainly is cause for wonder. We are at the cusp of rapid, unimaginable advances in machine learning — Google's self-learning algorithm trained for just four hours and smashed the world's top carbon-based Go players[2], neural nets are now writing the news[3], and it seems like with each day, tasks once thought of as requiring a uniquely human touch are usurped by an algorithm fresh out of Berkeley. Researchers have developed dizzying algorithms that depend on more computations than we can do in our lifetimes, and our processors are only getting faster. It seems like our foot is firmly on the gas pedal, and the sky is the limit.

But I'm skeptical.

Take AlphaGo, Google's aforementioned champion. In a 2019 panel on the limits of artificial intelligence, Oren Etzioni expounded on the remarkable accomplishments of AlphaGo, the landmark it represents, and the immense technical challenges that supported this feat. But then he paused, with a sly smile, and followed up with a simple question: "Now think, what has it done for us *recently*?"

[1] From documentary *Do You Trust This Computer*. This article has the highlights, including Elon's tweet that competition for AI superiority will cause WWIII.

[2] You can read more here.

The audience laughed, but his words weren't in jest. We have gotten really, really good at building algorithms to perform extremely specific tasks, but thats all they will ever be able to do; these algorithms don't learn or generalize in a holistic sense, they merely respond to an input incentive structure and tailor their mechanism to exploit it. This is the fundamental limitation of so much of our AI; progress is granular to the point of being insulated, and immense development can only catalyze niche application. And sure, maybe we're far from any type of general learning, but when it comes to leveraging our strengths to exponentially expand our corpus of capability, we can certainly get started.

The next great wave of machine learning lies in expanding algorithms to accommodate shifting tasks, representations, data, and assumptions.

One might even call it transfer learning.

[3]OpenAI made a fake news generator it deemed too dangerous to release.

Roadmap

This thesis is about transfer learning, the different ways we can represent it, and the bounds of its success. In Part I, I will conduct a survey over transfer learning, discussing its history, definition, and importance. In Part II, I introduce the mathematical machinery I will use to conduct my proofs, namely, the Machine Learning as Search Framework [Montañez, 2017]. I will also discuss what success means within this framework, and the various results we've been able to generalize.

In Part III, I present a number of theorems about transfer learning, original research from the Walter Bradley Center clinic team at Harvey Mudd College. This work was done by Tyler Sam, Abel Tadesse, Jake Williams, and myself, with advisors Dr. George Montañez and Dr. Robert J. Marks. I conclude this chapter with discussion of ongoing experimental work, and the path ahead.

Our research is made possible by the generous sponsorship of the Walter Bradley Center, a subsidiary of the Discovery Institute. We thank them for their guidance and support.

Chapter 1

Transfer Learning

Imagine you are the CEO of a car company, and you've decided to develop a self-driving car to compete with Tesla. You probably wouldn't start off by testing it on the road — there's the obvious public safety issue, and perhaps more important to your investors, it would cost a fortune to repair your prototype after every crash.

Instead, you might want to train the algorithm that controls your car within a simulation. It's important to note that this simulation environment and the real world are bound to be different: maybe there's an awkward roundabout in real life that doesn't show up in the simulation, or you purposefully didn't program in shadows because they would have made your simulation too slow. In any case, because they aren't identical, your algorithm (the self-driving car) needs a way of transferring what it learned in the simulation context to the real world.

This is the intuition behind transfer learning. At a high level, it's the process of taking knowledge gained by solving one problem and applying it to a different, but related problem. One might wonder how this could warrant an entire thesis. I encourage one to read on.

1.1 A Brief History

Like much of machine learning, transfer learning finds its roots in the theory of human cognition, taking concepts and terminology from psychology and philosophy. However at this point, a seasoned reader might think: "Bollocks! You've gone on about transfer learning, but you haven't defined it yet!"

This is true. One way to read this thesis is to skip forward to the mathematical definitions, read about the Framework (which earns that capital f), and scrutinize the proofs in our theorems to understand our results. I welcome such a reader. However, I anticipate that my audience might be a little more reluctant — perhaps I pleaded with you to read my work, and am currently peering over your shoulder, or you are humoring me out of good will up until the first lemma. To you, thank you for your attention (the first proof will appear on page 15), and I hope you walk away feeling like you have read a something like a story. And so like all great stories, we must start from the beginning[1].

1.1.1 Philosophical Roots

The limits of machination have captured the imagination of Western philosophy since Locke. Significant thought has been given towards the mechanistic concept of learning and the philosophy of mind, which has surprising relevance to modern computing.

John Searle, in his 1980 paper *Minds, Brains, and Programs*, famously argued for the theoretical shortcomings of AI with his Chinese Room hypothetical. Imagine that you've been trapped in a room by a manic mastermind, with a stack of tiles inscribed with Chinese characters[2], a lengthy manual, and enough Soylent to keep you going indefinitely. There are two slots in the corner of your favorite wall, one where character tiles are passed in, and one where you can pass tiles out. The manual serves as a guide: for any input sequence or combination of tiles, it has the information required to figure out which characters to pass back.

Now imagine that after a month, you get to be quite good at this (and rather fond of Soylent, you poor soul). And let's say that your responses, when relayed to the world outside, make complete logical sense to those who understand Chinese — in fact, you're being used as a successful (and lucrative) Justin Bieber chat bot. Searle claims that no matter how fast you respond or how sensible your responses are, you still fundamentally do not understand Chinese. In fact, since you are merely manipulating characters without grasping their meaning, you could be responding with complete nonsense and be none the wiser. Thus, no matter how many tiles you pass

[1] Sorry, Tarantino

[2] Alternatively, any language you don't understand

through the gap, you'll never actually *know* a single word of Chinese.

As you might expect, Searle then claims that artificial intelligence, no matter its sophistication, is analogous to you in the Chinese Room. At the end of the day, AI can be reduced to a step-by-step process or algorithm, and as such, it can never *learn* the latent meaning behind any concepts or structures no matter how fast or clever it gets.

This hypothetical has generated a great deal of critical attention. There have been a number of responses that are compelling to varying degrees: some philosophers compare the person in the Chinese Room to the physical processing unit of a computer instead of AI in general, others use the similarities between robots and children learning from sensory information as counter-evidence, and Pollock wrote at length about when semantics and syntax might come together.

I think this stuff is absolutely fascinating. Unfortunately, it is also far beyond the scope of this thesis. When I spoke about "learning concepts" in the Forward, my idea of "learning" is not the strict philosophical definition, but rather a colloquial one: for example, if some AI is able to predict my heartbeat at any given time-point of a Lakers game, I consider it as having learned this ability, regardless of the fact that it is merely doing some curve fitting and has no idea what a heartbeat is, or how it feels to watch Lakers superstar Alex Caruso throw down a baseline tomahawk.

Thus, the work in my thesis is not contingent on Searle's Chinese Room one way or another, nor does it depend on Fodor's Frame Problem or Gödel's Incompleteness Theorem. The limits that I explore are mathematical in nature and involve proofs and bounds instead of philosophy; my work addresses the limits under which transfer learning can be applied and the limits on how successful transfer can be, not whether machines can learn in the philosophical sense. One might wonder if I could have achieved this same point with just this paragraph, omitting mention of Chinese Rooms and Soylent.

One will never know.

1.2 Definition

Defining transfer learning turns out to be deceptively tricky. Not only is there disagreement on what counts as transfer learning, it is often difficult to cover its separate sub-disciplines under one umbrella definition. For the purpose of this thesis, I will present a general definition that encompasses

the majority of relevant sub-cases.

The most authoritative source on this subject is the 2010 paper *A Survey on Transfer Learning* [Pan and Yang, 2010], which has been a foundational resource in guiding our research efforts. In our work, I have modified some of their terminology for clarity (naming things is probably 70 percent of the fun of research), but the core definitions and relationships that Pan and Yang have defined remain the same.

1.2.1 Groundwork

Before we can talk about the definition of transfer learning, we must first introduce the idea of **domains** and **tasks**.

Imagine a machine learning problem where you are trying to predict whether someone likes capsicums based on their height and age. The domain is a way of representing the data that we are working with.

Definition 1.1 *Domain*

A domain $D = \{\mathcal{X}, P(X)\}$*, where* $\mathcal{X}$ *is a feature space and* $P(X)$ *is a marginal probability distribution.*

Let's try to arrive at this definition by thinking about what we need to describe our data. Typically, we think of our data as a collection of individual examples, which have a set of features and a label. In our case, one example might be {5'7, 21, Yes}, which corresponds to the format {height (feet), age (years), likes capsicum? (yes/no)}. If we focus on the height feature in our examples, we might expect that there are a lot more people who are 5'7 than 7'0. Thus, we could quite naturally represent the height data as a distribution over a one-dimensional space of heights.

Now, let's expand this to include both the height and age data. Each example corresponds to a point in 2-dimensional space, so we can consider the data as a whole as a distribution over a 2-dimensional feature space. Thus, $\mathcal{X}$ is this 2-dimensional space, and $P(X)$ is the aformentioned distribution, which together make up the domain.

Now that we've defined what a domain is, we can talk about tasks. A task can be thought of as the problem that we are trying to solve.

Definition 1.2 *Task*

Given domain D*, a task* $T = \{Y, f(\cdot)\}$*, where* Y *is a label space and* $f(\cdot)$ *is an objective predictive function.*

While it is tempting to say that our task is to try and predict whether people like capsicums, we are really only trying to predict whether some height/age combination will correspond to the label yes or no. Thus, our task is defined with respect to a certain domain — in particular, we are trying to learn what label to predict for any point in our domain. This mapping is the objective predictive function, and what are are mapping to is the label space. If we want to think of this probabilistically, this function can be represented as $P(Y|X)$, since we are trying to learn the probability of a label for a given point our domain.

1.2.2 Transfer Learning

Finally, we arrive at our definition of Transfer Learning.

Definition 1.3 *Transfer Learning*

Given a source domain $\mathcal{D}_S$ and learning task $\mathcal{T}_S$, as well as a recipient domain $\mathcal{D}_R$ and learning task $\mathcal{T}_R$, transfer learning aims to help improve the learning of the recipient predictive function $f_R(\cdot)$ in $\mathcal{D}_R$ using the knowledge in $\mathcal{D}_S$ and $\mathcal{D}_R$, where $\mathcal{D}_S \neq \mathcal{D}_R$ or $\mathcal{T}_S \neq \mathcal{T}_R$.

For an example of $\mathcal{D}_S \neq \mathcal{D}_R$, let's revisit our example of self-driving cars. Since the domain consists of both a feature space and a probability distribution, changing either will result in a different domain! Thus, applying Claremont driving data to New York City is a transfer learning problem where $\mathcal{D}_S \neq \mathcal{D}_R$, even though the features (types of data collected) may be the same!

For an example of $\mathcal{T}_S \neq \mathcal{T}_R$, imagine taking Claremont driving data and having two different tasks – maybe you start off training your algorithm to determine when to brake, and you want to use it to determine when to accelerate. This is a transfer learning problem where the domain is fixed but the task changes.

Finally, it can be the case that both $\mathcal{D}_S \neq \mathcal{D}_R$ and $\mathcal{T}_S \neq \mathcal{T}_R$. This is the classic self-driving car problem from the introduction: we want transfer what we learn from simulated driving and train an algorithm to drive in the real world. Note that while both tasks are driving-related, they are inherently different: the former task is to drive within the simulation and the latter is to drive in the real world – while these tasks are hopefully similar, there is no guarantee that they must be.

1.3 Current Applications

Thankfully, transfer learning has lucrative applications in the modern AI landscape, which pulls pressure off my prose to keep people reading our papers. In a 2016 NuerIPS conference, Andrew Ng, a prominent machine learning researcher and co-founder of Google Brain, discussed the exponentially increasing applications of transfer learning in industry.

I've mentioned one such application in self-driving cars, but we can actually expand this to include all simulation-based learning, where the simulation context is deviating from the real world in some way. Transfer learning also has applications in language — home assistants such as Alexa or Google need to understand a variety of accents and languages, and will transfer what they learn from accents they recognize to unfamiliar, new accents. Additionally, a popular approach in modern image classification is to take a model that was pre-trained on a large corpus of data such as Imagenet and strip off the top layer to serve as your starting point, rather than starting from scratch with randomized weights. This type of pre-training is a widely-used form of transfer learning.

The primary advantage of transfer learning, and what I suspect will be its primary utility in industry, is its ability to be flexible with data. Our machine learning models have gotten incredibly good in recent years — just take a look at AI-generated speech or video — but they require a vast amount of data, and often generalize poorly to conditions that deviate even slightly from what they are used to. Effective transfer can help with generalization, and more importantly, prevent the need for vast repositories of expensive, clean data.

1.4 The Problem at Hand

Now that we understand what transfer learning is, let's think back to our original example with the self-driving car. How well do you think transfer learning works?

If you are asking people to get in the back seat of an AI-driven supercar, transfer learning better be pretty damn good. And often times, if the simulation context is close enough to the real world, it probably will be. However, you might imagine that if the simulation was something like MarioKart and we were deploying in New York City, we might not do quite as hot when we

put the car on the road.

Our intuition suggests that the closer the two problems or tasks, the better transfer learning will be. One goal of our research is to find a representation of our problems that can express this difference, which must be flexible enough to apply to any problem, whether it involves cute animal pictures or MarioKart.

With this representation, another goal of our research is to find a way to say something about the efficacy of transfer learning given a fixed degree of similarity. In other words, if our problems or tasks have x amount of similarity by some metric, what can we say about the best possible performance of transfer learning? Are there conditions under which transfer learning is more likely to succeed? Conditions where it is sure to fail?

We think this is a really exciting problem. The current landscape of machine learning is like the 3am view from an airplane soaring across the U.S. — there's bright, punctuated clusters of progress in areas with ample funding and supply of data, but vast darkness in the space in between. With transfer learning, we can harness the work that's been done to slowly develop its neighboring regions, working until we're awash in light.

Or you know, until the robots get us.[3]

[3]This is a joke. :-)

Chapter 2

The Search Framework

As it turns out, going about proving theorems that hold for all of machine learning is quite a difficult task. Tremendous effort and capital is dedicated towards improving individual algorithms by incremental amounts, which dramatically diverge depending on the target problem and underlying assumptions. Even within a single subclass of machine learning, such as classification or regression, there can be great variation in the structure and approach of different algorithms.

To develop results that hold for all sorts of problems and algorithms, we need some way to talk about their commonalities. To do this, we use the Search Framework [Montañez, 2017]. For our purposes, a framework is simply a set of rules and definitions — if we show that a problem can be construed to fit within these rules, we can *cast* the problem into the framework. Through this process of casting, we can analyze a multitude of problems under the same structure or formalism, and apply theorems derived within the framework to each individual problem. The Search Framework, in particular, is a collection of rules and definitions that (perhaps unsurprisingly) make up a type of search, which allows us to use concepts from information theory to prove results about the problems we are interested in.

In this chapter, I will define the Search Framework, cast some common machine learning problems into the framework, and discuss some of the bounds and impossibility results that have been proven.

2.1 Definition

There are three essential components to a search problem. Imagine that you're on Netflix, looking for the perfect show to procrastinate writing your thesis to. There are a few different things you need (besides popcorn).

First, we need a **search space** Ω, which is what we are looking through in our search — in our example, Ω would be Netflix's entire catalog. Second, we need a **target** T, which is the subset of Ω that we're interested in, or the things that we are searching for. In our case, this might consist of the shows in Netflix's catalog that we think are sufficiently distracting.

Lastly, we need a way to evaluate how good a show is. We call this our **external information resource** F, which gives us feedback on any given element of our search space. In our example, F might be something like RottenTomatoes. For a successful search, we expect there to be a tight relationship between our information resource and the target set: if your definition of a good movie aligns with RottenTomatoes ratings, finding a good movie to watch with our method seeming promising, but it may very well be that you find RottenTomatoes to be pretentious and inaccurate[1], in which case we would expect the search to do poorly.

To describe this with terminology, let ω be an element in Ω (a specific movie Netflix's catalog), or the null element (where we start). Then, the evaluation of ω by our information resource is denoted $F(\omega)$.[2]

In summary, a search problem is as follows:

Definition 2.1 *Search Problem*

A search problem is defined as a three-tuple (Ω, T, F)*, consisting of a search space, a target subset of the space, and an external information resource, respectively.*

2.1.1 Algorithmic Search

Now that we have an idea of what a search problem is, let's consider what a process that solves this problem, an algorithmic search, would look like.

Let's be clear on what this means. The following algorithm is not a specific approach to a particular task, rather, it represents a general algorithmic

[1] National Treasure got a dismal 46% rating, which is considered rotten. The audacity.

[2] This is a slight abuse of terminology. To be more accurate, we apply an extraction function g to our information resource to get our evaluation, so we define $F(\omega) := g(F, \omega)$

framework of sorts that uses a level of abstraction to encompass all search algorithms. With this, we can reason about any of the algorithms we use to solve the problem we cast into the framework, and make some interesting comments regarding their shared properties.

The black-box search algorithm is as follows:

Algorithm 1 Black-box Search Algorithm

Initialize $h_0 \leftarrow (\emptyset, F(\emptyset))$.
for all $i = 1, \ldots, i_{\max}$ **do**
    Using history $h_{0:i-1}$, compute P_i, the distribution over Ω.
    Sample element ω according to P_i.
    Set $h_i \leftarrow (\omega, F(\omega))$.
end for
if an element of T is contained in any tuple of h **then**
    Return *success*.
else
    Return *failure*.
end if

We define h as the history of an algorithm, which contains tuples $(\omega, F(\omega))$ at each step h_i. We initialize the history with the null element and our evaluation of it. Then, for a predetermined number of steps, our algorithm generates a distribution over the search space, we sample an element from the distribution, and append the element and our evaluation to our history. The process of generating the distribution over our search space is abstracted as a sort of black box, and represents our algorithm's best guess at the target. After the final iteration, if any member of our search history was in the target set, we consider the search a success. Otherwise, we consider it a failure.

Let's consider what this approach looks like with our Netflix example. At each step of the algorithm, we use the search history (titles we examined and their RottenTomato ratings) to generate a distribution over the catalog, which we then sample from to get the next title. While we abstract away the details that go into making this distribution, one could imagine a number of different approaches that would work. One feasible algorithm might generate a distribution with non-zero probability on movies that feature directors or actors from highly rated titles in our history. Another might put all the probability mass on the next title in alphabetical order. Not all algorithms will be effective: consider one that generates a uniform distribution over the

search space, regardless of the history. When we sample from this distribution at each step, we are essentially choosing a random title in the catalog to examine — not exactly the winningest strategy.

Note that this approach encapsulates both deterministic algorithms and algorithms that incorporate some sort of chance. In the deterministic case, an algorithm generates a distribution with all the probability mass on a single element, which is then chosen trivially when we sample from said distribution. In the stochastic case, we can explain the element of chance through the inherent randomness in sampling from distributions; if we sample from a uniform distribution at each step, for example, it is clear that running the algorithm multiple times will likely result in different outcomes.

Here is a figure to illustrate this process more colorfully:

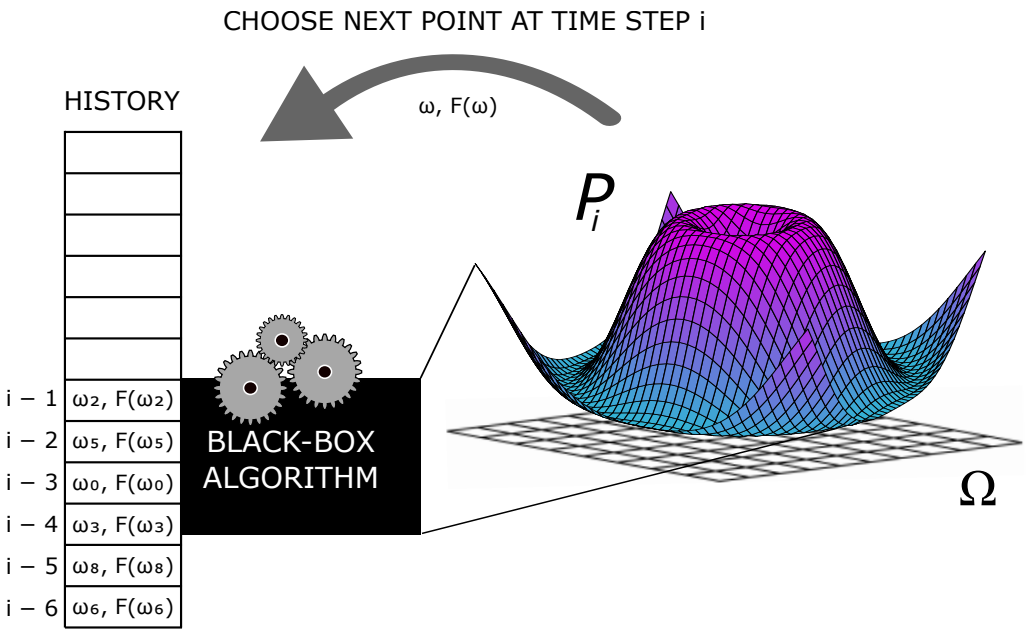

Figure 2.1: Black-box search algorithm. We iteratively populate the history with samples from a distribution that is determined by the black-box at each iteration, using the history [Montañez, 2017].

2.2 Machine Learning as Search

At this point, we've developed a framework, and shown how it describes search problems such as our Netflix example. While we think this is pretty neat, a framework is just a bunch of rules and a catchy name — for it to be useful, we have to show that many different types of machine learning problems can be cast into it, or in other words, show how a variety of problems can be seen as search problems within the framework.

2.2.1 Regression as Search

Let's start by considering the case of simple linear regression, the task of estimating the relationship between a dependant variable and an independent variable. More colloquially, this is the task of fitting a line to some two-dimensional data.

Since lines can be defined by their slope and intercept, we can let the search space Ω be the set of all possible (slope, intercept) pairs, and the target T be the subset of these pairs that are acceptable by some criteria. For example, T can be the set of lines that minimize the sum of squares below some value ϵ. Then, our information resource F will just be the training data and loss function, which together allow us to evaluate any given line in Ω.

To accommodate the case of general regression without the assumption of linearity or in higher dimensions, we can simply expand our search space Ω to include all possible regression functions, keeping our definitions of T and F. As classification can be reduced to a special case of regression, we can imagine a similar definition.

2.2.2 Self-Driving Cars as Search

In my introduction to transfer learning, I brought up self-driving cars as an example of a machine learning problem that I'd like to analyze. However, this problem is a lot more complex and multifaceted than simply regression or classification — self-driving algorithms feature a multitude of intersecting components that analyze input data, make predictions about future conditions, adjust different settings, and so on.

Thus, rather than breaking down each component of a specific algorithm and describing how the different parts may fit into the framework independently, for our analysis, I will imagine the self-driving car algorithm as a

black box with a bunch of knobs and dials that correspond to the weights and parameters that make up its components. For our car to drive well, we need to search through all the possible configurations of these dials (Ω) for the ones that preform well (T). We can evaluate how well our algorithm is doing by collecting metrics on driving data F.

2.2.3 More Problems as Search

Along with regression, Montañez proved that multiple different types of problems, such as classification, clustering, parameter estimation, and Vapnik's General Learning Problem, can be cast into the framework [Montañez, 2017].

2.2.4 Algorithmic Success

Now that we have a sense of what individual algorithms look like within the search framework, we need a metric for the probability of success of an algorithm in order to compare them. Montañez introduced the expected per-query probability of success as a way to compare algorithms that may involve a feature of randomness, and proved a host of impossibility results and bounds with this definition.

Definition 2.2 *Expected Per-Query Probability of Success Metric*

$$q(t, f) = \mathbb{E}_{\tilde{P},H}\left[\frac{1}{|\tilde{P}|}\sum_{i=1}^{|\tilde{P}|} P_i(w \in t)\middle| f\right] = \overline{P}(X \in t|f)$$

where $\tilde{P}$ is the sequence of probability distributions generated by the black box, H is the search history, and t and f are the target set and information resource of the search problem, respectively [Montañez, 2017]

We define the expected per-query probability of success of an algorithm with respect to a fixed target and information resource. To handle variations in the search history that may result from sampling, or variations in the way our black box constructs each probability distribution P_i, our evaluation of success must be in expectation over all possible sources of randomness. For a given search history, made up of a sequence of probability distributions P_i, we want to average the probability of selecting an element of our target set at each step.

As we began working within this framework, however, we realized that the expected per-query probability of success metric is insufficient to characterize the case of transfer learning, where we care more about the transferred state, i.e. the last few iterations of the algorithm, than the expected average success over all the steps. This drove us to define another probability of success metric, the general probability of success.

Definition 2.3 *General Probability of Success Metric*

$$q_\alpha(t, f) = \mathbb{E}_{\tilde{P},H}\left[\sum_{i=1}^{|\tilde{P}|} \alpha_i P_i(w \in t)\Big| f\right] = P_\alpha(X \in t|f)$$

where P_α is a valid probability distribution on the search space Ω and α_i is the weight allocated to the ith probability distribution in our sequence [Sam et al., 2020]

Now, instead of weighting the probability of success of each individual step P_i equally, we can select the amount of probability mass to place on each step. Thus, we can see that the expected per-query probability of success is an instance of the general probability of success where each α_i is $\frac{1}{|\tilde{P}|}$. However, instead of generalizing these theorems for the general probability of success, we realized that the most important property of the the expected per-query probability of success was that it could be represented as a linear function of a probability distribution. We defined this property formally as decomposability, showed that both the expected per-query probability of success and general probability of success are decomposable, and generalized our theorem for any decomposable probability-of-success metric[Sam et al., 2020].

Definition 2.4 *Decomposability*

*A probability-of-success metric ϕ is **decomposable** if and only if there exists a $\mathbf{P}_{\phi,f}$ such that*

$$\phi(t, f) = \mathbf{t}^\top \mathbf{P}_{\phi,f} = P_\phi(X \in t|f).$$

The next section contains a number of theorems we generalized using decomposable probability-of-success metrics. As the general probability of success allows us to consider any well-defined weighting of our steps, this extends our theorems to hold **regardless of how you define success**.

2.3 Generalizing Search Framework Theorems

There have been a number of interesting results proven within the Search Framework. Using our new property of decomposability, we can generalize many existing theorems for broader application.

2.3.1 No Free Lunch

A general version of No Free Lunch [Wolpert and Macready, 1997] can be proven within the framework, which states that algorithmic performance is conserved, or in other words, doing better at one task necessarily implies doing worse at another [Sam et al., 2020].

Theorem 2.5 (No Free Lunch for Search and Machine Learning) *For any pair of search/learning algorithms $\mathcal{A}_1$, $\mathcal{A}_2$ operating on discrete finite search space Ω, any closed under permutation set of target sets τ, any set of information resources $\mathcal{B}$, and decomposable probability-of-success metric ϕ,*

$$\sum_{t\in\tau}\sum_{f\in\mathcal{B}}\phi_{\mathcal{A}_1}(t,f)=\sum_{t\in\tau}\sum_{f\in\mathcal{B}}\phi_{\mathcal{A}_2}(t,f).$$

Proof Note that the closed under permutation condition implies $\sum_{\mathbf{t}\in\tau}\mathbf{t}=[c,c,\ldots,c]=\mathbf{1}\cdot c$ for some constant c.

$$\begin{aligned}
\sum_{t\in\tau}\sum_{f\in\mathcal{B}}\phi_{\mathcal{A}_1}(t,f) &= \sum_{\mathbf{t}\in\tau}\sum_{f\in\mathcal{B}}\mathbf{P}^{\top}_{\phi,f,\mathcal{A}_1}\mathbf{t}\\
&= \sum_{f\in\mathcal{B}}\mathbf{P}^{\top}_{\phi,f,\mathcal{A}_1}\sum_{\mathbf{t}\in\tau}\mathbf{t}\\
&= \sum_{f\in\mathcal{B}}\mathbf{P}^{\top}_{\phi,f,\mathcal{A}_1}\mathbf{1}\cdot c\\
&= c\sum_{f\in\mathcal{B}}\mathbf{P}^{\top}_{\phi,f,\mathcal{A}_1}\mathbf{1}\\
&= c\sum_{f\in\mathcal{B}}1\\
&= c\sum_{f\in\mathcal{B}}\mathbf{P}^{\top}_{\phi,f,\mathcal{A}_2}\mathbf{1}
\end{aligned}$$

$$= \sum_{\mathbf{t} \in \tau} \sum_{f \in \mathcal{B}} \mathbf{P}^{\top}_{\phi, f, \mathcal{A}_2} \mathbf{t}$$
$$= \sum_{t \in \tau} \sum_{f \in \mathcal{B}} \phi_{\mathcal{A}_2}(t, f)$$

∎

This proof highlights the importance of decomposability: by decomposing the success metric into a dot product between a probability distribution and target vector, we can manipulate the outer summation and represent the target vector in terms of some constant c. After pulling c out of the summation, we can reduce further by observing that the dot product between a probability vector and the identity vector must be 1, as probability distributions must sum to 1. We can then reverse this process to substitute in another algorithm, yielding our result.

The straightforward interpretation of this result is that no algorithm can preform better than any other algorithm over all closed-under-permutation target sets. This implies that, if a fixed algorithm has an increased performance on some target, information resource pair, it must come at the cost of performance elsewhere, or in other words, success is conserved.

Let's think back to our self-driving car example. If our algorithm gets very good at driving under icy conditions, applying that configuration of knobs and settings to another context, such as the suburbs of Claremont, will likely make our car drive worse. However, one might point out that there must certainly be some self-driving car algorithm that is smart enough to get better at maneuvering in both contexts. This is definitely true — while this theorem states that improving at certain target sets necessitates doing worse on others, it doesn't dictate what these target sets actually are. With the abundance of possible target sets, it's often not a bad thing to trade performance — as an example, if our algorithm is getting better at specializing, it will certainly get worse at generalizing by definition, but we're okay with that. This theorem is not suggesting that all algorithms perform the same given some fixed target, or that they are not useful, it simply makes an observation about conservation over the entire, vast space of targets.

2.3.2 Famine of Favorable Targets

While the No Free Lunch theorem uses the breadth of possible targets to make a claim about overall algorithmic performance, we can also fix an

algorithm and information resource to prove results about the scarcity of favorable targets in different ways [Sam et al., 2020].

Theorem 2.6 (The Famine of Favorable Targets) *For fixed $k \in \mathbb{N}$ (representing a fixed number of targets in each target vector), fixed information resource f, and decomposable probability-of-success metric ϕ, define*

$$\begin{aligned}\tau &= \{T \mid T \subseteq \Omega, |T| = k\}, \text{ and} \\ \tau_{q_{min}} &= \{T \mid T \subseteq \Omega, |T| = k, \phi(T, f) \geq q_{min}\}.\end{aligned}$$

Then,

$$\frac{|\tau_{q_{min}}|}{|\tau|} \leq \frac{p}{q_{min}}$$

where $p = \frac{k}{|\Omega|}$.

Proof Let $\mathcal{S} = \{\mathbf{s} : \mathbf{s} \in \{0,1\}^{|\Omega|}, \|\mathbf{s}\| = \sqrt{k}\}$. For brevity, we will allow $\mathbf{s}$ to also denote its corresponding target set, letting the context make clear whether the target set or target function is meant. Then,

$$\begin{aligned}\frac{|\tau_{q_{\min}}|}{|\tau|} &= \frac{\sum_{\mathbf{s}\in\mathcal{S}} \mathbb{1}_{\phi(\mathbf{s},f)\geq q_{\min}}}{\binom{|\Omega|}{k}} \\ &= \binom{|\Omega|}{k}^{-1} \sum_{\mathbf{s}\in\mathcal{S}} \mathbb{1}_{\phi(\mathbf{s},f)\geq q_{\min}} \\ &= \mathbb{E}_{\mathcal{U}[\mathcal{S}]}\left[\mathbb{1}_{\phi(\mathbf{S},f)\geq q_{\min}}\right] \\ &= \Pr(\phi(\mathbf{S}, f) \geq q_{\min}) \\ &\leq \frac{\mathbb{E}_{\mathcal{U}[\mathcal{S}]}[\phi(\mathbf{S}, f)]}{q_{\min}},\end{aligned}$$

where the final step follows from Markov's inequality. By decomposability of ϕ and linearity of expectation, we have

$$\begin{aligned}\frac{\mathbb{E}_{\mathcal{U}[\mathcal{S}]}[\phi(\mathbf{S}, f)]}{q_{\min}} &= \frac{\mathbb{E}_{\mathcal{U}[\mathcal{S}]}[\mathbf{S}^\top \mathbf{P}_{\phi,f}]}{q_{\min}} \\ &= \frac{\mathbf{P}_{\phi,f}^\top \mathbb{E}_{\mathcal{U}[\mathcal{S}]}[\mathbf{S}]}{q_{\min}}\end{aligned}$$

$$
\begin{aligned}
&= \frac{\mathbf{P}_{\phi,f}^{\top}\mathbf{1}\left[\binom{|\Omega|}{k}^{-1}\binom{|\Omega|-1}{k-1}\right]}{q_{\min}} \\
&= \frac{k}{|\Omega|}\frac{\mathbf{P}_{\phi,f}^{\top}\mathbf{1}}{q_{\min}} \\
&= \frac{p}{q_{\min}}.
\end{aligned}
$$

∎

This theorem shows that our threshold of success $q_{\min}$ is inversely proportional to the cardinality of satisfying target sets. For example, if we want our car to drive really well, the set of satisfying configurations for our algorithm will be really low. However, if we are okay with our car driving just okay, the set of satisfying configurations will be much larger. We can get intuition by thinking back to our black-box representation of the self-driving car algorithm. If we don't care very much about how well we do, turning a few knobs on our black box won't matter too much, but if we want our algorithm to preform very well, messing around with the knobs could be catastrophic.

2.3.3 Fraction of Favorable Targets

We can also discuss the relative performance of an algorithm in terms of bits. In Montañez's work, "relative performance" was defined in comparison to the benchmark of uniform random sampling, measured by the expected per-query probability of success [Montañez, 2017]. However, we can generalize this theorem for any decomposable probability-of-success metric by defining a version of active information of expectations for decomposable metrics $I_{\phi(t,f)}$, which represents the relative advantage of our algorithm in bits [Sam et al., 2020]. However, before I do so, I will introduce two lemmata we will use to simplify our final bound.

Lemma 2.7 (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 [Sauer, 1972] for completeness.

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

$$
\begin{aligned}
&\leq \left(\frac{n}{d}\right)^d \sum_{j=0}^{n} \binom{n}{j} \left(\frac{d}{n}\right)^j \\
&= \left(\frac{n}{d}\right)^d \left(1+\frac{d}{n}\right)^n \\
&\leq \left(\frac{n}{d}\right)^d \lim_{n\to\infty} \left(1+\frac{d}{n}\right)^n \\
&= \left(\frac{en}{d}\right)^d.
\end{aligned}
$$

∎

This lemma gives us an upper bound for a combinatorial summation in terms of e. We can use this in the following lemma to derive another useful bound.

Lemma 2.8 (Binomial Approximation) $\sum_{j=0}^{\lfloor \frac{n}{2^b} \rfloor} \binom{n}{j} \leq 2^{n-b}$ *for* $b \geq 3$ *and* $n \geq 2^b$.

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

$$2b + \log_2 e \leq 2^b$$

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

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

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

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

Thus,

$$
\begin{aligned}
2^n &\geq 2^{b + \frac{n}{2^b}(b+\log_2 e)} \\
&= 2^b 2^{\frac{n}{2^b}(b+\log_2 e)}
\end{aligned}
$$

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

where the penultimate inequality follows from the Sauer-Shelah inequality [Sauer, 1972]. Dividing through by 2^b gives the desired result. ∎

Now that we have some useful bounds under our belt, let's return to the task at hand. We are interested in quantifying the advantage an algorithm has over the baseline of uniform random sampling. Thus, we define the active information of expectations for decomposable metrics $I_{\phi(t,f)} := -\log_2 \frac{p}{\phi(t,f)}$, where $p = |t|/|\Omega|$, the baseline per-query probability of success for uniform random sampling with replacement. Essentially, $I_{\phi(t,f)}$ is the ratio of the success of our algorithm to the baseline in bits, which allows us to quantify the relative advantage of our algorithm [Sam et al., 2020].

Theorem 2.9 (The Fraction of Favorable Targets) *Let* $\tau = \{t \mid t \subseteq \Omega\}$, $\tau_b = \{t \mid \emptyset \neq t \subseteq \Omega, I_{\phi(t,f)} \geq b\}$, *and decomposable probability-of-success metric* ϕ. *Then for* $b \geq 3$,

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

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

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

For $|\Omega| < 2^b$ and $I_{\phi(T,F)} \geq b$, 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 2.8, we have

$$\begin{aligned}\frac{|\tau_b|}{|\tau|} &\leq \frac{|\tau'_b|}{|\tau|} \\ &= 2^{-|\Omega|} \sum_{k=0}^{\lfloor \frac{|\Omega|}{2^b} \rfloor} \binom{|\Omega|}{k} \\ &\leq 2^{-|\Omega|} 2^{|\Omega|-b} \\ &= 2^{-b}.\end{aligned}$$

∎

This result provides an exponentially decreasing bound on the fraction of favorable targets with respect to the magnitude of advantage. The larger an advantage we want, the smaller the satisfying target set will be, demonstrating the scarcity of favorable b-bit target vectors. While the self-driving car application is similar to the previous theorem — increasing how well we want to drive decreases the number of satisfying target sets — this result shows the connection between the Search Framework and information-theoretic representations such as bits.

2.4 Generalizing Bias

Imagine that you're in the mood for bananas. You head over to the local farmer's market, where you've heard there's a varied spread of bananas from local farms, with the objective of picking the tastiest bananas from the bunch. There are a few different strategies you can use to choose your bananas — for example, if you randomly select bananas from the produce cart, you are preforming a type of unbiased search. However, you might recall from previous experiences that yellow bananas are more likely to taste good than green bananas or spotted bananas, and introduce some bias into your search by keeping an eye out for color. This biased search will likely result in a higher proportion of tasty bananas in your final yield than an unbiased search, demonstrating the value of using knowledge about the target to influence your search [Montañez et al., 2019].

As you might imagine, this notion of bias is incredibly important if we want our algorithms to succeed. However, it's important to draw the distinction between this technical definition and the way bias is usually referred

to in machine learning. To be clear, this bias does not refer to the harmful social biases that can be encoded into algorithms by unrepresentative or discriminatory data, but rather the bias that allows algorithms to generalize beyond their training distribution. If an algorithm has no bias and assumes nothing, any arbitrary pattern in the data can be possible, so the only thing it can do is guess randomly to make predictions about the future.

Montañez et al. defines bias in the search framework as a measure of the extent a distribution of information resources is predisposed to a fixed target [Montañez et al., 2019].

Definition 2.10 *Bias in the Search Framework*

For a distribution $\mathcal{D}$ over a collection of possible information resources $\mathcal{F}$, with $F \sim \mathcal{D}$, and a fixed k-hot target $\boldsymbol{t}$, Bias is defined as:

$$\begin{aligned} Bias(\mathcal{D}, \boldsymbol{t}) &= \mathbb{E}_{\mathcal{D}}[\boldsymbol{t}^\top \overline{\mathbf{P}}_F] - \frac{k}{|\Omega|} \\ &= \boldsymbol{t}^\top \mathbb{E}_{\mathcal{D}}[\overline{\mathbf{P}}_F] - \frac{\|\boldsymbol{t}\|^2}{|\Omega|} \\ &= \boldsymbol{t}^\top \int_{\mathcal{F}} \overline{\mathbf{P}}_f \mathcal{D}(f)\, df - \frac{\|\boldsymbol{t}\|^2}{|\Omega|}. \end{aligned}$$

In words, the bias between the distribution and the target is defined as the difference between the expected average performance of an algorithm and the baseline of uniform random sampling.

We can re-define bias in terms of any given decomposable probability-of-success metric $\phi(T, F)$ by substituting $\mathbf{P}_{\phi,F}$ for $\overline{\mathbf{P}}_F$ [Sam et al., 2020].

Definition 2.11 *Generalized Bias*

$$\begin{aligned} Bias_\phi(\mathcal{D}, \boldsymbol{t}) &= \mathbb{E}_{\mathcal{D}}[\boldsymbol{t}^\top \boldsymbol{P}_{\phi,F}] - \frac{k}{|\Omega|}. \\ &= \boldsymbol{t}^\top \mathbb{E}_{\mathcal{D}}[\boldsymbol{P}_{\phi,F}] - \frac{\|\boldsymbol{t}\|^2}{|\Omega|} \\ &= \boldsymbol{t}^\top \int_{\mathcal{F}} \boldsymbol{P}_{\phi,f} \mathcal{D}(f)\, df - \frac{\|\boldsymbol{t}\|^2}{|\Omega|}. \end{aligned}$$

This generalized bias allows us to consider different ways an algorithm can be predisposed towards a target, and prove some interesting results.

2.4.1 Futility of Bias-Free Search

While I alluded to this earlier, we can actually prove that algorithms are unable to perform better than uniform random sampling without bias. With our generalized definition of bias, we show that this holds however we measure the predisposition of our information resources to a target [Sam et al., 2020].

Theorem 2.12 (Futility of Bias-Free Search) *For any fixed algorithm $\mathcal{A}$, fixed target $t \subseteq \Omega$ with corresponding target function $\mathbf{t}$, and distribution over information resources $\mathcal{D}$, if $\text{Bias}_\phi(\mathcal{D}, \mathbf{t}) = 0$, then*

$$\Pr(\omega \in t; \mathcal{A}) = p$$

where $\Pr(\omega \in t; \mathcal{A})$ represents the expected decomposable probability of successfully sampling an element of t using $\mathcal{A}$, marginalized over information resources $F \sim \mathcal{D}$, and p is the single-query probability of success under uniform random sampling.

Proof Let $\mathcal{F}$ be the space of possible information resources. Then,

$$\begin{aligned}\Pr(\omega \in t; \mathcal{A}) &= \int_{\mathcal{F}} \Pr(\omega \in t, f; \mathcal{A}) \mathrm{d}f \\ &= \int_{\mathcal{F}} \Pr(\omega \in t \mid f; \mathcal{A}) \Pr(f) \mathrm{d}f.\end{aligned}$$

Since we are considering the per-query probability of success for algorithm $\mathcal{A}$ on t using information resource f, we have

$$\Pr(\omega \in t \mid f; \mathcal{A}) = P_\phi(\omega \in t \mid f).$$

Also note that $\Pr(f) = \mathcal{D}(f)$ by the fact that $F \sim \mathcal{D}$. Making these substitutions, we obtain

$$\begin{aligned}\Pr(\omega \in t; \mathcal{A}) &= \int_{\mathcal{F}} P_\phi(\omega \in t \mid f) \mathcal{D}(f) \mathrm{d}f \\ &= \mathbb{E}_{\mathcal{D}} \left[P_\phi(\omega \in t \mid F) \right] \\ &= \mathbb{E}_{\mathcal{D}} \left[\mathbf{t}^\top \mathbf{P}_{\phi, F} \right] \\ &= \text{Bias}_\phi(\mathcal{D}, \boldsymbol{t}) + p \\ &= p.\end{aligned}$$

∎

This result demonstrates the futility of a search devoid of bias. If we are trying to train a self-driving car algorithm and the driving data isn't predisposed towards some outcome, the algorithm cannot do better than randomly guessing what to do. For example, imagine training your algorithm on driving data where the car flips a coin at traffic lights to decide whether to stop or go, regardless of the color of the light — and every decision results in the same outcome (not crashing). In this case, the best any algorithm can do is guess randomly at what to do! Our data is not biased towards any outcome, so our algorithm has no way of learning the right response. On the other hand, if the information resource has high bias towards some outcome, and crashes whenever the car drives through a red light, our car will have a much higher probability of driving better (if it's still in one piece).

2.4.2 Famine of Favorable Information Resources

While we've previously generalized results about the relative sparsity of favorable target sets, we can use our notion of bias to generalize results about information resources as well. [Montañez et al., 2019].

Theorem 2.13 (Famine of Favorable Information Resources) *Let $\mathcal{B}$ be a finite set of information resources and let $t \subseteq \Omega$ be an arbitrary fixed k-size target set with corresponding target function $\mathbf{t}$. Define*

$$\mathcal{B}_{q_{\min}} = \{f \mid f \in \mathcal{B}, \phi(t, f) \geq q_{\min}\},$$

where $\phi(t, f)$ is an arbitrary decomposable probability-of-success metric for algorithm $\mathcal{A}$ on search problem (Ω, t, f), and $q_{\min} \in (0, 1]$ represents the minimally acceptable probability of success. Then,

$$\frac{|\mathcal{B}_{q_{\min}}|}{|\mathcal{B}|} \leq \frac{p + \text{Bias}_{\phi}(\mathcal{B}, \mathbf{t})}{q_{\min}}$$

where $p = \frac{k}{|\Omega|}$.

Proof We seek to bound the proportion of successful search problems for which $\phi(t, f) \geq q_{\min}$ for any threshold $q_{\min} \in (0, 1]$. Let $F \sim \mathcal{U}[\mathcal{B}]$. Then,

$$\frac{|\mathcal{B}_{q_{\min}}|}{|\mathcal{B}|} = \frac{1}{|\mathcal{B}|} \sum_{f \in \mathcal{B}} \mathbb{1}_{\phi(t,f) \geq q_{\min}}$$

$$= \mathbb{E}_{\mathcal{U}[\mathcal{B}]}[\mathbb{1}_{\phi(t,F)\geq q_{\min}}]$$
$$= \Pr(\phi(t,F) \geq q_{\min}).$$

By decomposability, we have

$$\frac{|\mathcal{B}_{q_{\min}}|}{|\mathcal{B}|} = \Pr(\mathbf{t}^\top \mathbf{P}_{\phi,F} \geq q_{\min}).$$

Applying Markov's Inequality and by the definition of $\text{Bias}(\mathcal{B}, \mathbf{t})$, we obtain

$$\frac{|\mathcal{B}_{q_{\min}}|}{|\mathcal{B}|} \leq \frac{\mathbb{E}_{\mathcal{U}[\mathcal{B}]}[\mathbf{t}^\top \mathbf{P}_{\phi,F}]}{q_{\min}}$$
$$= \frac{p + \text{Bias}_\phi(\mathcal{B}, \mathbf{t})}{q_{\min}}.$$

∎

This result demonstrates that we need our set of information resources to be biased to our target in order to have a high proportion of favorable information resources, or information resources that lead to a high probability of success. Additionally, if we fix the bias between a distribution of information resources and a target, increasing our threshold of success corresponds inversely to the size of the set of satisfying information resources. Going back to the self-driving car example, the degree to which a distribution of training data is predisposed towards some outcome is related to the degree to which any sampled set of data will yield a high probability of success.

Chapter 3

Transfer Learning Results

Now that we're well-acquainted with the Search Framework and the type of results we can prove within it, let's narrow our focus to the case of transfer learning.

3.1 Transfer Learning as Search

Recall that transfer learning involves two separate machine learning problems, a source problem and recipient (target) problem. For the most part, we can cast these problems into our framework separately, and analyze them independently.

However, a central premise of transfer learning is that we can use information we learn from the source problem to help us in the recipient problem. To express this within our framework, we introduce a notion of learned information L, which can be encoded as some finite-length binary string. L is generated from our source problem, and is passed to the recipient problem to supplement its initial information resource f_{R_o}. With this representation, we can prove a number of interesting results about transfer learning.

3.2 Affinity

To study how the recipient problem improves from the information it gains, we need to consider a version of bias that deals with this learned knowledge L. As we saw in the previous chapter, bias is defined with respect to a distribution of information resources and a target. However, our case carries

additional complexity, as our recipient information resource has two components, the initial information resource f_{R_o} and the learned information L. Thus, we want to study a notion of bias where our distribution of information resources is such that each information resource contains f_{R_o}, and the variability is expressed entirely through L. To express this, we define $\mathcal{D}_L$ to be a distribution placed over the possible learning resources L, which is generated from the source problem, and define an analogue of bias called affinity.

Definition 3.1 *Affinity*

Consider a transfer learning problem with a fixed k-hot target $\boldsymbol{t}$*, fixed recipient information resource* f_{R_o}*, and a distribution* $\mathcal{D}_L$ *over a collection of possible learning resources* $\mathcal{L}$*, with* $L \sim \mathcal{D}$*. The* ***affinity*** *between the distribution and the recipient problem is defined as:*

$$\begin{aligned} \mathit{Affin}(\mathcal{D}_L, \boldsymbol{t}, f_{R_o}) &= \mathbb{E}_{\mathcal{D}_L}[\boldsymbol{t}^\top \mathbf{P}_{\phi,L,f_{R_o}}] - \boldsymbol{t}^\top \mathbf{P}_{\phi,f_{R_o}} \\ &= \boldsymbol{t}^\top \mathbb{E}_{\mathcal{D}_L}[\mathbf{P}_{\phi,L,f_{R_o}}] - \phi(t, f_{R_o}) \\ &= \boldsymbol{t}^\top \int_{\mathcal{L}} \mathbf{P}_{\phi,L,f_{R_o}} \mathcal{D}_L(l)\, dl - \phi(t, f_{R_o}) \end{aligned}$$

To build intuition about affinity, consider the ways it differs from the generalized bias we defined earlier. While our baseline for bias is the probability of success of random uniform sampling, the baseline for affinity is $\phi(t, f_{R_o})$, the success of an algorithm with just the initial information resource and no learned information. The nastier term $\mathbb{E}_{\mathcal{D}_L}[\mathbf{t}^\top \mathbf{P}_{\phi,L,f_{R_o}}]$ simply represents the expected probability of success over the distribution of information resources that include the learned resource.

Thus, affinity essentially measures the extent to which the distribution of learned information generated by the source problem is successful at augmenting the recipient problem's original information resource. It seems intuitive that, for transfer learning to be successful, affinity should be high.

3.3 Transfer Learning Results

With affinity, we can prove results similar to our bias results in the previous chapter, which give us insight to the ways the probability of success of transfer learning relates to the information in the source and recipient problems.

3.3.1 Futility of Affinity-Free Search

To begin, we can prove that without affinity, our transfer learning can do no better than the baseline success of the recipient algorithm.

Theorem 3.2 (Futility of Affinity-Free Search) *For any fixed algorithm $\mathcal{A}$, fixed recipient problem (t, f_{R_o}), where $t \subseteq \Omega$ with corresponding target function $\mathbf{t}$, and distribution over information resources $\mathcal{D}_L$, if* $\text{Affin}(\mathcal{D}_L, \mathbf{t}) = 0$*, then*

$$\begin{aligned}\Pr(\omega \in t; \mathcal{A}_L) &= \Pr(\omega \in t; \mathcal{A}) \\ &= \phi(t, f_{R_o})\end{aligned}$$

where $\Pr(\omega \in t; \mathcal{A}_L)$ represents the expected decomposable probability of successfully sampling an element of t using $\mathcal{A}$, marginalized over learning resources $L \sim \mathcal{D}_L$, and $\phi(\boldsymbol{t}, f_{R_o})$ is the probability of success without L under the given decomposable metric.

Proof Let $\mathcal{L}$ be the space of possible learning resources. Then,

$$\begin{aligned}\Pr(\omega \in t; \mathcal{A}_L) &= \int_{\mathcal{L}} \Pr(\omega \in t, l; \mathcal{A}) \mathrm{d}l \\ &= \int_{\mathcal{L}} \Pr(\omega \in t \mid l; \mathcal{A}) \Pr(l) \mathrm{d}l.\end{aligned}$$

Since we are considering the per-query probability of success for algorithm $\mathcal{A}$ on t using learning resource l, with a fixed recipient information resource f_{R_o}, the following are equivalent:

$$\Pr(\omega \in t \mid l; \mathcal{A}) = P_{\phi, f_{R_o}}(\omega \in t \mid l) = P_{\phi, f_{R_o}, l}(\omega \in t).$$

Also note that $\Pr(l) = \mathcal{D}_L(l)$ because our information resources are drawn from the distribution $\mathcal{D}_L$. Making these substitutions, we obtain

$$\begin{aligned}\Pr(\omega \in t; \mathcal{A}_L) &= \int_{\mathcal{L}} P_{\phi, f_{R_o}, l}(\omega \in t) \mathcal{D}_L(l) \mathrm{d}l \\ &= \mathbb{E}_{\mathcal{D}_L}\left[P_{\phi, f_{R_o}, L}(\omega \in t)\right] \\ &= \mathbb{E}_{\mathcal{D}_L}\left[\mathbf{t}^\top \mathbf{P}_{\phi, L, f_{R_o}}\right] \\ &= \text{Affin}(\mathcal{D}_L, \boldsymbol{t}) + \mathbf{t}^\top \mathbf{P}_{\phi, f_{R_o}}\end{aligned}$$

$$= \phi(t, f_{R_o}).$$

∎

The interpretation of this is similar to the analogous theorem for bias. If the distribution of learned information we transfer is not predisposed to the target, we can't expect to do any better than the baseline success of the recipient problem. In the self-driving car example, if we're transferring learned information from a simulation context to the real world, but the information we learned doesn't help us identify ways to drive better, this transfer will not benefit us when we start training our self-driving car on the streets.

3.3.2 Famine of Favorable Learned Information Resources

With affinity, we can also prove a similar bound on the proportion of favorable learned information resources.

Theorem 3.3 (Famine of Favorable Learned Information Resources) *Let L be a finite set of learning resources and let $t \subseteq \Omega_R$ be an arbitrary fixed target set of size k. Given a recipient problem, (t, f_{R_o}). Define*

$$L_{q_{\min}} = \{l \mid l \in L, \phi(t, f_{R_o+l}) \geq q_{\min}\},$$

where $\phi(t, f_{R_o+l})$ is the decomposable probability-of-success metric for algorithm $\mathcal{A}$ on search problem (Ω, t, f_{R_o+l}) and $q_{\min} \in (0, 1]$ represents the minimally acceptable per-query probability of success. Then,

$$\frac{|L_{q_{\min}}|}{|L|} \leq \frac{\phi(t, f_{R_o}) + \text{Affin}(D_L, \mathbf{t}, f_{R_o})}{q_{\min}}$$

where $\phi(t, f_{R_O})$ is the decomposable probability-of-success metric with the recipient's original information resource.

Proof We seek to bound the proportion of successful search problems for which $\phi(t, f) \geq q_{\min}$ for any threshold $q_{\min} \in [0, 1]$. Let $F \sim \mathcal{U}[L]$. Then,

$$\frac{|L_{q_{\min}}|}{|L|} = \frac{1}{|L|} \sum_{l \in L} \mathbb{1}_{\phi(t, f_{R_o+l}) \geq q_{\min}}$$

$$= \mathbb{E}_{\mathcal{U}[L]}[\mathbb{1}_{\phi(t, f_{R_o+l}) \geq q_{\min}}]$$
$$= \Pr(\phi(t, f_{R_o+L}) \geq q_{\min}).$$

Let $\omega \in t$ imply that the target function $\boldsymbol{t}$ evaluated at ω is one. Then, by the definition of decomposability,

$$\frac{|L_{q_{\min}}|}{|L|} = \Pr(\boldsymbol{t}^\top \boldsymbol{P}_{\phi, f_{R_o+L}} \geq q_{\min}).$$

Since $D_L \sim \mathcal{U}[L]$, we can attribute all the randomness in F_{R_o+L} to the learned information, L. Then, applying Markov's Inequality and by the definition of $\text{Affin}(\mathcal{A}, \boldsymbol{t})$,

$$\frac{|L_{q_{\min}}|}{|L|} \leq \frac{\mathbb{E}_{\mathcal{U}[L]}[\boldsymbol{t}^\top \boldsymbol{P}_{\phi, L, f_{R_o}}]}{q_{\min}}$$
$$= \frac{\phi(t, f_{R_o}) + \text{Affin}(D_L, \boldsymbol{t}, f_{R_o})}{q_{\min}}.$$

∎

Thus, the affinity of a distribution of learned information resources correlates directly to the proportion of useful learned information. In the self-driving car example, the more a distribution of learned information from the simulation tells us about the real world, the higher the proportion of useful learned information there will be.

3.4 Applied Results

While we've been able to prove some interesting theoretical results about transfer learning, the theorems in this chapter thus far function primarily as theoretical backing to ideas that are quite intuitive and simple. The remainder of this chapter will be dedicated to applied results, where I will revisit the questions I posed in the introduction — how can we compare the similarity of two arbitrary problems, and how do we use this similarity to construct a bound on the efficacy of transfer?

3.4.1 A Bound on Transfer Learning

Since our similarity metric must be flexible to accommodate arbitrary problems (and since we're already working within a search framework), a natural

solution is to use information-theoretic concepts such as mutual information to express the similarity between problems. Mutual information can be thought of as the degree to which two variables depend on each other. It is formally defined as the reduction in uncertainty (entropy) of a random variable after seeing another random variable — this is essentially just the amount of information one variable tells us about another — in units of shannons, or bits.

Using the mutual information between the information resource of the source problem and the recipient target set, we can construct an upper bound on the probability of success of transfer learning. This proof follows the Probability of Success Under Dependence Theorem [Montañez, 2017], which is a reversed generalization of Fano's Inequality [Fano and Hawkins, 1961].

Theorem 3.4 (Learning under Dependence for Transfer Learning) *Define* $\tau_k = \{T_R \mid T_R \subseteq \Omega_R, |T_R| = k \in \mathbb{N}\}$ *and let* $\mathcal{B}_m$ *denote any set of binary strings (information resources), such that the strings are of length* m *or less. Define* q_{TL} *as the expected decomposable probability of success under the joint distribution on* $T_R \in \tau_k$ *and* $F_{R_0+L} \in \mathcal{B}_m$ *for any fixed algorithm* $\mathcal{A}$*, such that* $q_{TL} := \mathbb{E}_{T_R, F_{R_0+L}}[\phi(T_R, F_{R_0+L})]$*, namely,*

$$q_{TL} = \mathbb{E}_{T_R, F_{R_0+L}}[\, P_\phi(\omega \in T_R | F_{R_0+L})] = \Pr(\omega \in T_R; \mathcal{A}).$$

Then,

$$q_{TL} \leq \frac{I(F_S; T_R) + I(F_{R_o}; T_R) + D(P_{T_R} \| \mathcal{U}_{T_R}) + 1}{I_\Omega}$$

where $I_\Omega = -\log k/|\Omega_R|$*,* $D(P_{T_R} \| \mathcal{U}_{T_R})$ *is the Kullback-Liebler divergence between the marginal distribution on* T_R *and the uniform distribution on* T_R*, and* $I(F;T)$ *represents mutual information between an information resource and target.*

Proof Let $Z = \mathbb{1}(\omega \in T_R)$. We can use the chain rule for entropy to expand $H(Z, T_R|\omega)$ in two different ways:

$$\begin{aligned} H(Z, T_R|\omega) &= H(Z|T_R, \omega) + H(T|\omega) \\ &= H(T_R|Z, \omega) + H(Z|\omega). \end{aligned}$$

By definition of Z, $H(Z|T_R, \omega) = 0$, and by the data processing inequality, $H(T_R|F_{R_o+L}) \leq H(T_R|\omega)$. Thus,

$$H(T_R|F_{R_o+L}) \leq H(T_R|Z, \omega) + H(Z|\omega).$$

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

$$\begin{aligned} H(T_R|Z,\omega) &= (1-P_g)H(T_R|Z=0,\omega) + P_g H(T_R|Z=1,\omega) \\ &\leq (1-P_g)\binom{|\Omega|}{k} + P_g\binom{|\Omega|-1}{k-1} \\ &= \binom{|\Omega|}{k} - P_g\frac{|\Omega|}{k}. \end{aligned}$$

Let $H(\mathcal{U}_{T_R}) = \binom{|\Omega|}{k}$ be the entropy of the uniform distribution over k-sparse target sets in Ω_R. Substituting, we get

$$H(T_R|F_{R_o+L}) \leq H(\mathcal{U}_{T_R}) - P_g\frac{|\Omega|}{k} + H(Z|\omega).$$

Using the definitions of conditional entropy and I_Ω, we get

$$H(T_R) - I(T_R; F_{R_o+L}) \leq H(\mathcal{U}_{T_R}) - P_g I_\Omega + H(Z|\omega),$$

which implies

$$\begin{aligned} P_g I_\Omega &\leq I(T_R; F_{R_o+L}) + H(\mathcal{U}_{T_R}) - H(T_R) + H(Z|\omega) \\ &= I(T_R; F_{R_o+L}) + D(P_{T_R}\|\mathcal{U}_{T_R}) + H(Z|\omega). \end{aligned}$$

Now, let's analyze the term $H(Z|\omega)$. This term captures the amount of entropy in Z we can attribute to the randomness in the target set T_R. As a visualization, we can imagine ω as our selected bet in the overall roulette wheel Ω [Montañez, 2017]. From our discussion on the process of algorithmic search, recall that we "chose" elements of our target set when the ball drops on a random slot, which is determined by the distribution induced on T_R. $H(Z|\omega)$ is maximized when the ball randomly drops on ω, or in other words, when we choose ω as often as we do not, about half the time. Thus, this entropy term captures the contribution of luck inherent in the randomness of the sampling process, averaged over all ω, and can be upper-bounded by its maximum value of 1. When we start to select ω each time instead of just randomly, we know can attribute it to something other than luck. Thus,

$$\Pr(\omega \in T_R; \mathcal{A}) \leq \frac{I(T_R; F_{R_o+L}) + D(P_{T_R}\|\mathcal{U}_{T_R}) + 1}{I_\Omega}$$

Furthermore, we can substitute q_{TL} for $\Pr(\omega \in T_R; \mathcal{A})$. Note that in our definition $q_{TL} = \mathbb{E}_{T_R, F_{R_o+L}}[\, P_\phi(\omega \in T_R|F_{R_o+L})]$, we can be assured that q_{TL}

is a proper probability distribution, as expectation has properties of linearity and boundedness.

Finally, we can bound the mutual information between our updated information resource and the recipient target in terms of the similarity between the source and recipient information resources.

Define $I(F_{R_o+L}; T_R) := I(F_{R_o}, L; T_R)$. Then,

$$\begin{aligned}
I(T_R, F_{R_o+L}) &= I(F_{R_o+L}, T_R) \\
&= I(F_{R_o}, L; T_R) \\
&= I(L; T_R \mid F_{R_o}) + I(F_{R_o}; T_R) \\
&= H(L \mid F_{R_o}) - H(L \mid F_{R_o}, T_R) + I(F_{R_o}; T_R) \\
&= H(L \mid F_{R_o}) - H(L \mid T_R) + I(F_{R_o}; T_R) \\
&\leq H(L) - H(L \mid T_R) + I(F_{R_o}; T_R) \\
&= I(L; T_R) + I(F_{R_o}; T_R) \\
&\leq I(F_S; T_R) + I(F_{R_o}; T_R)
\end{aligned}$$

To arrive at this inequality, we use the symmetry of mutual information to swap the order of the terms and plug in our definition of the updated learning resource F_{R_o+L}. We then apply the chain rule, apply the definition of mutual information, use the property of conditional independence, bound the entropy, re-apply the definition of mutual information, and apply the data processing inequality.

Thus, we can represent our bound in terms of the source and recipient by substituting in the above inequality, which yields:

$$q_{TL} \leq \frac{I(F_S; T_R) + I(F_{R_o}; T_R) + D(P_{T_R} \| \mathcal{U}_{T_R}) + 1}{I_\Omega}$$

∎

This theorem provides an upper bound for the probability of successful transfer learning q_{TL}, indicating that transfer learning can only help us up to a certain point, with that threshold being determined largely by the mutual information between the source information resource and recipient target, $I(F_S; T_R)$. This makes a lot of sense, as this value encodes the amount of reliable information we receive about the location of the target of the recipient task, from the information resource of the source problem. In concrete

terms, thinking back to our self-driving car example, this is the mutual dependence between the simulation data and the optimal configurations of the self-driving car algorithm, which describes the amount and the usefulness of the information we can extract and thereby transfer from the simulation.

The probability of success also depends on the mutual information between the initial information resource and target of the recipient $I(F_{R_o+L}; T_R)$. This term captures the contribution of exploitable information that our initial driving data offers. If we're applying the algorithm trained on the simulated data to the real world directly (as data may be expensive or difficult to obtain), this contribution may very well be 0. $D(P_{T_R} \| \mathcal{U}_{T_R})$ is the extent to which the distribution over the recipient targets diverges from the uniform distribution — a small value indicates similarity to the uniform, which leaves us at the whims of randomness (the uniform distribution maximizes entropy), while a large value indicates a strong divergence from the uniform distribution, implying the likelihood of the existence of only a few target sets. Thus, the KL-Divergence can be interpreted as the predictability of the recipient target sets, or the extent to which the optimal configurations of our self-driving algorithm can be identified.

Finally, the addition of 1 serves as an upper bound against the natural randomness inherent in a search process, and I_Ω is defined to be the cost of finding an element of the recipient target set without any additional information. Thus, this theorem connects the difficulty of the recipient problem, the value of the transferred information, luck of the draw, and the predictability of our target to bound the probability of success of transfer learning.

3.4.2 Experimental Similarity Metric

We're excited about our theoretical bound, but its practical applications are limited for two reasons: first, it's immensely difficult to calculate terms such as KL-Divergence and mutual information in complex machine-learning problems[1], second and slightly more damning, it requires knowledge of the target set to calculate in the first place.

Thus, we've pivoted our research to explore similarity metrics experimentally, in an attempt to discover simple heuristics that indicate conditions in which transfer learning may be successful. Given a source problem and recip-

[1]In fact, we are exploring the use of experimentally determined q_{TL} values to bound mutual information! Quite the reverse.

ient problem, one such heuristic we explored was the success of an algorithm on the recipient problem after training solely on the source problem and **not** the recipient problem.

To test this, we focused on two similar image classification problems, classifying tigers versus wolves[2] (TvW) and classifying cats versus dogs[3] (CvD). Due to the parallels in these two problem, we expect that a model trained for one task will be able to help us with the other. In our experiment, we used a generic image classification model (VGG16)[4] to evaluate the aforementioned heuristic to see whether it correlates with any benefit in transfer learning. The table below contains our experimental data:

Run	Source Problem	Testing Accuracy	Recipient Problem	Additional Training?	Testing Accuracy
1	CvD	84.8%	TvW	N	74.24%
2	CvD	84.8%	TvW	Y	95.35%
3	TvW	92.16%	CvD	N	48.36%
4	TvW	92.16%	CvD	Y	82.44%

The `Source Problem` column denotes the problem we are transferring from, and the `Recipient Problem` column denotes the problem we are transferring to. The first `Testing Accuracy` column contains the image classification model's testing accuracy on the source problem after training on its dataset. The `Additional Training?` column indicates whether we did any additional training before testing the model's accuracy on the recipient problem's dataset — `N` indicates no training, which means that the following entry in the second `Testing Accuracy` column contains the results of the heuristic, while `Y` indicates an additional training phase, which means that the following entry in the second `Testing Accuracy` column contains the experimental performance of transfer learning. No matter what, for each run, we start by training our model on the source problem.

Let's start by analyzing Runs 1 and 2. Run 1 is the heuristic run for the CvD $\rightarrow$ TvW transfer learning problem. When we apply the trained CvD model to the TvW problem without retraining, we get a testing accuracy of 74.24%. This result is promising, as it's significantly above a random

[2] `http://image-net.org/challenges/LSVRC/2014/browse-synsets`

[3] `https://www.kaggle.com/c/dogs-vs-cats-redux-kernels-edition/data`

[4] `https://keras.io/applications/#vgg16`

50% coin flip, indicating that our CvD model has learned something about the difference between cats and dogs that can be weakly generalized to other images of feline and canine animals. Looking at Run 2, we see that taking our model and training additionally on the TvW dataset yields a transfer learning testing accuracy of 95.35%, which is higher than the testing accuracy when we train our model solely on TvW (92.16%)! This is an example where transfer learning improves our model's success, suggesting that the pre-training step is helping our algorithm combat overfitting.

When we look at Runs 3 and 4, we see the other side of the picture. The heuristic for the TvW $\rightarrow$ CvD transfer learning problem in Run 3 is a miserable 48.36%, which is how well we would do randomly flipping a coin. It's important to note that this heuristic is not symmetric, which is to be expected — for example, if the TvW model is learning based on the background of the images and not the animals themselves, we would expect a poor application to the CvD problem regardless of how well the CvD model can apply to the TvD problem. Looking at Run 4, the transfer learning testing accuracy is 82.44%, which is below the testing accuracy when we train solely on the CvD dataset (84.8%). This offers support to our heuristic — when the success of our heuristic is closer to random, it seems like pre-training not only fails to benefit our algorithm, but can even hurt our performance.

With these results in mind, let's interpret our heuristic. A high value means that the algorithm trained on the source problem is able to preform well on the recipient problem, which indicates that the algorithm is able to identify and discriminate between salient features of the recipient problem. Thus, when we transfer what it learns (the model weights), we expect to see a boost in performance. Conversely, a low value (around 50%, much lower and we can flip the labels) indicates that the algorithm is unable to learn features useful for the recipient problem, so we would expect transfer to be unsuccessful. It's important to note that this heuristic is heavily algorithm dependent, which differs from our theoretical results — problems with a large degree of latent similarity can receive poor values by our heuristic if the algorithm struggles to learn the underlying features of the problem.

These results provide a promising proof of concept for our heuristic. However, a lot more experimentation needs to be done across a wide array of transfer learning problems before anything conclusive can be said. Our work is ongoing, so fingers crossed for a paper in 2021!

3.5 A Look Back, Next Steps

In this thesis, I've presented an overview of the work the Walter Bradley Center clinic team has done over the past year. We started with a thorough and arduous literature review on transfer learning, and began our experimental work by building out machinery within the Search Framework to prove results about transfer learning. Along the way, we defined decomposability and generalized a number of theorems about search and bias, among which a selected few were presented in this thesis. We eventually compiled our work into an article (our research advisor has a saying — if it's not in Tex, it didn't happen!) and the team went off to Malta to present our research[5] at ICAART 2019.

Upon our return, we focused on casting transfer learning into the Search Framework, which led us to define affinity and prove a host of related theorems. It took multiple rounds of iteration to develop a satisfactory representation of transfer learning, a process that was greatly abbreviated within this thesis (we started replacing names for Tex commands in our notes due to the high turnaround). In our current working representation, we achieved one of our initial goals of constructing an information theoretic bound for the success of transfer learning, presented earlier in this chapter.

We were excited about our work, but with the gentle prodding of our research advisors, decided to expand our direction to more practical application in an effort to excite others as well. We started by constructing toy problems to evaluate the tightness our bounds, and began ongoing experimental work to develop and evaluate different similarity heuristics. We also made an attempt to develop theoretical, geometric similarity metrics within the framework, but quickly found ourselves face to face with the curse of dimensionality. This led to some spirited, albeit ultimately tangential research sessions — but as a result of these times, we successfully petitioned to annex a neighboring floor-to-ceiling whiteboard, which became an invaluable asset, Wall-ter Bradley.

After COVID-19 separated us, we began to write our second paper, which will center around our work with affinity. We had tentative plans to create interactive visual representations of transfer learning with Rubik's Cubes written in JavaScript, and to extensively broaden our experimental work, but much of that has been put to the wayside as we deal with newfound

[5]It's allowed me to cite myself in this thesis, which is fun

challenges in each of our lives.

While there is no dearth of experimental exploration to conduct, the most compelling next steps for me are theoretical in nature. We've built out definitions and vocabulary in our framework to tackle transfer learning, but in many of our results, our goal of developing theorems that can be applied to many different types of problems has led to a lack of nuanced insight — the content in our proofs is almost more interesting than the results themselves. Along the same vein, we ran into issues when we tried constructing lower bounds for the success of transfer learning, because with no guarantees about the nature of the problems we transfer to, it's very difficult to say anything concrete — in our abstract construction, an algorithm can very well disregard all the learned information it's passed and randomly query the null element, an adversarial example our results need to be robust to.

Thus, the next avenue of research is to investigate theorems that rely on extensive assumptions within the framework, which trade off between applicability and generalizability. I don't think this is inherently a contradictory premise — the framework allows for many different problems to be discussed with the same general structure, so with some additional work, I think it's possible to identify general metrics that can selectively narrow our focus from the vast space of possible algorithms and targets to ones that are more interesting. I'm not sure what these metrics would look like, but I am confident in the creativity of the researchers to come.

This thesis has been a joy to write. I hope you have found it interesting. Machine learning and AI are terms often thrown around to invoke a sense of mathematical authority (and to be honest, obscurity), but at it's core, the concepts of learning and success are simple, intuitive, and at the risk eliciting groans, beautiful.

Thank you for reading.

Bibliography

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

[Kearns, 1989] Kearns, M. J. (1989). *Computational Complexity of Machine Learning*. PhD thesis, Department of Computer Science, Harvard University.

[Michalski et al., 1983] Michalski, R. S., Carbonell, J. G., and Mitchell, T. M., editors (1983). *Machine Learning: An Artificial Intelligence Approach, Vol. I*. Tioga, Palo Alto, CA.

[Mitchell, 1980] Mitchell, T. M. (1980). The need for biases in learning generalizations. Technical report, Computer Science Department, Rutgers University, New Brunswick, MA.

[Montañez, 2017] Montañez, G. D. (2017). The Famine of Forte: Few Search Problems Greatly Favor Your Algorithm. In *Systems, Man, and Cybernetics (SMC), 2017 IEEE International Conference on*, pages 477–482. IEEE.

[Montañez, 2017] Montañez, G. D. (2017). *Why Machine Learning Works*. PhD thesis, Carnegie Mellon University.

[Montañez et al., 2019] Montañez, G. D., Hayase, J., Lauw, J., Macias, D., Trikha, A., and Vendemiatti, J. (2019). The Futility of Bias-Free Learning and Search. *CoRR*, abs/1907.06010.

[Pan and Yang, 2010] Pan, S. J. and Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*.

[Sam et al., 2020] Sam, T., Williams, J., Tadesse, A., Sun, H., and Montañez, G. D. (2020). Decomposable probability-of-success metrics in algorithmic search.

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

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