CS181P Final Project Proposal #2 - Literature Review

Santi Santichaivekin, Jingyi Liu, Jacob Boerma, Lorraine Zhao, Princewill Okoroafor
Harvey Mudd College and Claremont McKenna College
Claremont, California, USA

1 Literature Review

So far, we have read the following documents. We summarize them below.

Why ML Works http://www.cs.cmu.edu/~gmontane/montanez_dissertation.pdf [Lorraine]

Famine of Forte https://arxiv.org/pdf/1609.08913.pdf [Jacob]

Probabilistic machine learning and artificial intelligence https://www.nature.com/articles/nature14541 [Rose]

Generalization as Search http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.121.5764&rep=rep1&type=pdf [Lorraine]

Data Compression Explained http://mattmahoney.net/dc/dce.html#Section_13 [Rose] [Princewill]

Minimum Description Length https://www.cs.cmu.edu/~aarti/Class/10704/lec13-MDL.pdf [Princewill]

No Free Lunch For Optimization https://ti.arc.nasa.gov/m/profile/dhw/papers/78.pdf [Santi] [Jacob]

Why ML Works

`http://www.cs.cmu.edu/~gmontane/montanez_dissertation.pdf` [Lorraine]

In George Montanezs work why Machine Learning works, chapter 4, he specifies that learning problems can be reduced into different search problems by various means.

First, this chapter talks about regression as search, where a function from a set of input a real valued output is learned. Given a dataset D that has both training instances with class labels and test set, and a external information resource, we can view each regression algorithm as a search that outputs a function according to the loss function from a large collection of possible regression functions.

Second, George mentions a specific case of the above exampleclassification as search, which is a type of regression with output values restricted to a small set, where class labels are often categorical.

Third, the paper examines clustering as search, where we have a set of points in a vector space and we need to cluster them into distinct groups with a fixed set of unknown clusters and a truth assignment of points to clusters. While soft-clustering algorithm allows a point to be grouped into more than one cluster, hard-clustering algorithm assigns each point to only one cluster. It is important to note that the better the vector space representation of the points, the more dependent between the external information source and the target set of acceptable clusterings; and that the better how the points are chosen (like sampling i.i.d.), the more informative the set of points concerning the true cluster assignments.

Fourth, the paper discusses parameter estimation as search, where we have a large search space of parameter vectors, each of which could be used as input to a black-box process. The better the restrictions are detailing how output data is produced with the true parameter vector, the more informative the external information source is concerning the true parameter vector.

Fifth, the paper continues to discuss on hyperparameter optimization as search, where we have a sequential optimization procedure that samples elements from the search space, evaluates them according to the external information source and adds the feedback to the history. Sixth, the paper briefly describes general learning problem as search, where we can translate Vapniks generalization of learning into the search framework.

Famine of Forte

`https://arxiv.org/pdf/1609.08913.pdf` [Jacob]

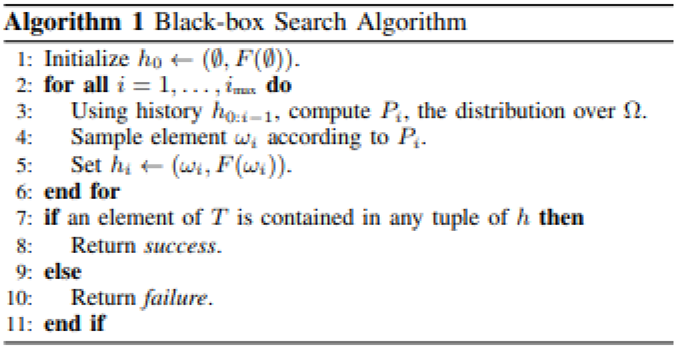
This paper uses a search framework to prove several bounds concerning the proportion of good problems for a search algorithm.

The paper uses a finite **search space** Ω, and objective is to find an element of the **target set** $T \subset \Omega$. Given an order of the search space Ω, we can represent T as a binary vector of size Ω, because each element of the search space corresponds to one index in our vector. This vector is the **target function**.

Finally, the paper describes an information resource F. Similar to the No Free Lunch for Optimization paper, Famine of Forte treats F as an oracle from which our algorithm gets all information about a particular point $\omega \in \Omega$. $F(\omega)$ is the evaluation of ω. Interestingly, this paper abstracts this resource as a finite bit string.

Given a search space Ω, a target set T, and an information resource F, a search problem is defined as (Ω, F, T). Since we are abstracting F and T to be finite-length bit strings, we can do lots of fun things!

The paper describes a **Black-box Search Algorithm**, that attempts to find points ω in the target set T. The algorithm keeps track of its history H, which is a series of tuples $h_i = (\omega_i, F(\omega_i))$. Each tuple keeps track of the ith call to the oracle. At iteration i, the algorithm uses the history up to this point to compute a distribution P_i over Ω. The algorithm chooses some element w_i according to the distribution and consults the oracle. Note especially that this algorithm will perform queries i_{max} times - only after does it check if we have seen a target element.



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 $\Omega$.
    Sample element $\omega_i$ according to $P_i$.
    Set $h_i \leftarrow (\omega_i, F(\omega_i))$.
end for
if an element of $T$ is contained in any tuple of $h$ then
    Return *success*.
else
    Return *failure*.
end if
```

We care about the performance of our search algorithm, since it is sufficiently general to represent many ML algorithms.

Since the algorithm performs i_{max} queries to the oracle, but this number can change, we may measure its performance by the expected per-query probability of success $q(T|F)$. The paper goes on to discuss how since this expectation is over all sources of randomness, it is equal to the probability of success for samples drawn from an appropriately averaged distribution - I don't really understand this.

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

The paper defines $p(T, F)$ to be the expected probability of success when performing uniform random sampling, and uses this a baseline for the performance of our algorithm. It may be worth checking out the following paper, since it is referenced by Famine of Forte when discussion $p(T, F)$.

W. Dembski and R. Marks II, Conservation of information in search: Measuring the cost of success, Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on, vol. 39, no. 5, pp. 1051 1061, sept. 2009

Theorem 1. Famine of Forte

This result shows that the proportion of search problems with performance $q(T, F) \geq q_{\text{min}}$ is bounded above. Formally, the paper defined a set τ_k, which is a set of k-hot target vectors. That is, τ_k contains all possible k-element target sets.

The paper restricts the potential information resources to be in the set B_m, where B_m is any set of binary strings such that each string has length m or less. The paper then defines sets R and $R_{q_{min}}$, where R contains all search problems with k-length target sets, and $R_{q_{min}}$ those problems that perform as well as q_{min}. Their definitions are below:

$$\begin{aligned} \tau_k &= \{T | T \subset Q, |T| = k \in \mathbb{N}\} \\ R &= \{(T, F) | T \in \tau_k, F \in B_m\} \\ R_{q_{min}} &= \{(T, F) | T \in \tau_k, F \in B_m, q(T, F) \geq q_{min}\} \end{aligned}$$

Then

$$\frac{|R_{q_{min}}|}{|R|} \leq \frac{p}{q_{min}} = \frac{k/|\Omega|}{q_{min}}$$

The same result holds as $m \to \infty$ - as the size of the information sets grows.

Essentially, we are given a search space Ω. In that space, the proportion of problems (Ω, F, T) in which our algorithm performs as well as q_{min} is bounded above by an exact quantity.

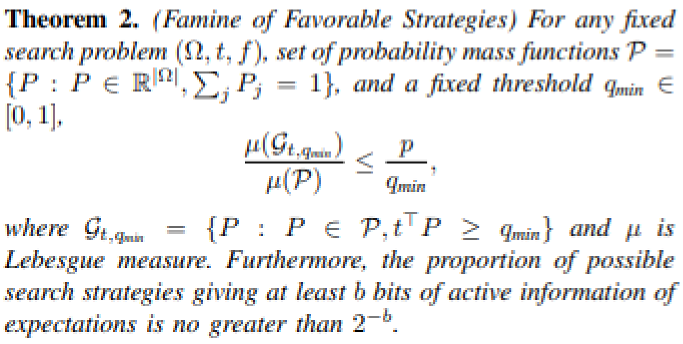
Corollary 1. (Conservation of Active Information of Expectations)

If we define $I_{q(T,F)} = -\log(p/q(T, F))$, and effectively use this metric as our performance (with minimum performance b), then we get the result

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

This uses the exact same set construction as theorem 1. Note that $I_{q(T,F)}$ looks very similar to the difference in surprisals of p and q - p and q are expected per-query probabilities of success. Our algorithm does better than uniform random sampling if I is large, since that indicates a higher per-query probability of success. Thus the proportion of problems that do as well as b under this metric is bounded above by 2^{-b}. The paper discusses how this shows that the improved search method is equivalent to a uniform random sampler on a reduced search space. (I'm not really sure how this works)

Theorem 2. Famine of Favorable Strategies



Theorem 2. *(Famine of Favorable Strategies) 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 $q_{min} \in [0, 1]$,*

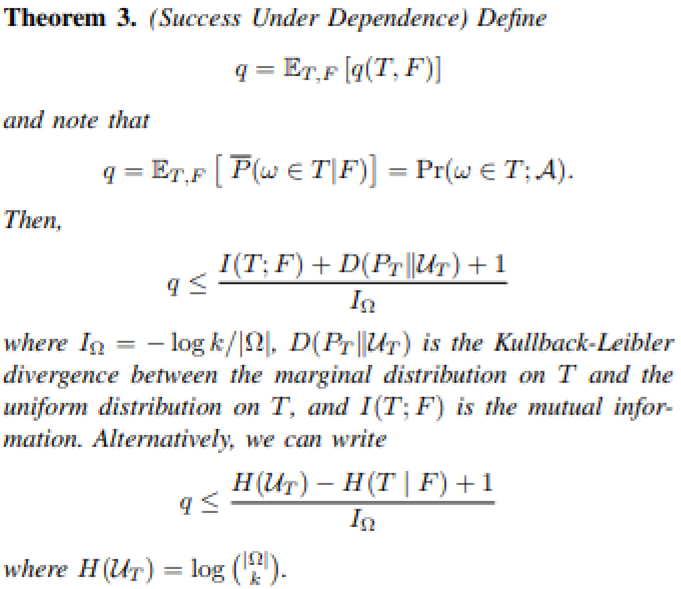
$$\frac{\mu(\mathcal{G}_{t,q_{min}})}{\mu(\mathcal{P})} \leq \frac{p}{q_{min}},$$

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}.

To be honest, I am not sure what this theorem is saying at the moment. My guess is that over all possible sequences $\{P_i\}_i$ of probabilities we can bound the number of algorithms that perform well on a given search problem.

The paper discusses below how this places bounds on the number of favorable strategies on a given problem. This result and theorem 1 show that favorable problems for strategies and favorable strategies for problems are rare.

Theorem 3. Success Under Dependence



Theorem 3. *(Success Under Dependence) Define*

$$q = \mathbb{E}_{T,F}\left[q(T, F)\right]$$

and note that

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

Then,

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

where $I_\Omega = -\log k/|\Omega|$, $D(P_T \| \mathcal{U}_T)$ is the Kullback-Leibler 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 $H(\mathcal{U}_T) = \log \binom{|\Omega|}{k}$.

This upper-bounds the expected performance of our algorithm on a search problem. The paper discusses how this bounds improves monotonically with the dependence between between our target set and information resources. That is, if our information is somewhat accurate, we can perform better!

Other than that statement, I am having a hard time understanding this theorem.

The paper then goes on to discuss examples of problems that fit under the framework, and proves theorem 1 for the space of hyperparameters for a given algorithm A. It then shows that there cannot be a fitness function (information resource) that works well for all target sets in a search space. In other words, there is no One-Size-Fits-All fitness function.

The paper then discusses why novel machine learning algorithms keep getting generated. Since the proportion of favorable strategies for a given problem is small and the proportion of favorable problems for a strategy is small, we must keep generating new strategies to deal with new problems.

Further Reading

J. Culberson, On the futility of blind search: An algorithmic view of no free lunch, Evolutionary Computation, vol. 6, no. 2, pp. 109127, 1998

** Just want to see no free lunch in another context

Compression and machine learning: a new perspective on feature space vectors

https://ieeexplore.ieee.org/abstract/document/1607268

** Compression - ML direction.

An Introduction to Kolmogorov Complexity and Its Applications - skim chapters 2 and 3

** I read some stuff on Kolmogorov complexity, which is a way of measuring how random a string is / how hard a string is to generate. The complexity of a string is the number of characters needed to specify it (can use a programming language, so long as we include the specification in our length). Some strings are hard to generate, others are easy. There seems to already be a no-free-lunch theorem for compression based on the pigeonhold principle, but Kolmogorov complexity might help to give bounds on the number of strings that can be greatly compressed in a given language.

Probabilistic machine learning and artificial intelligence

https://www.nature.com/articles/nature14541 [Rose]

In his paper Probabilistic machine learning and artificial intelligence, Ghahramani provides an overview of the probabilistic approach to machine learning and the state-of-art advances in this field. Specifically, he explores five areas that fall under this framework including probabilistic programming, Bayesian optimization, probabilistic data compression, automatic model discovery, and hierarchical modeling. Under the framework of probabilistic modeling, Ghahramani views machine learning as a process of inferring the most plausible model from a given data set, which is in turn used to improve performance on future predictions. He points out the advantages of using a probabilistic framework by arguing the central role of uncertainty in machine learning models and identifies three sources of uncertainty that are typically present in modeling (452-453). The uncertainties consist of the measurement noise of the data set, the unknown parameters in a given model, and uncertainties about the general structure of the mode. Thus probabilistic modeling comes as a natural remedy for dealing with various sources of uncertainties and identifies the best model in accordance with data. Aside from its ability to represent uncertainties in the model, probabilistic modeling also brings the advantage of flexibility through the use of Bayesian non-parametric models, which grow in complexity as the data set grows (454). Such flexibility allows machine learning models to capture the regularities of the data and have better performance. Interestingly, he points out that Bayesian nonparametrics is equivalent to a neural network with infinite hidden units. In the section about Bayesian optimization, he formulates the optimization process as a sequential decision problema search problem in our wordswhere you use a probabilistic distribution of the function values and determine the next query by choosing the area with the most uncertainty (and thus the most information gain once the value is reviewed) (456). Here we notice a connection between a specific type of machine learning problems and search is made through the bridge of the probabilistic framework. In the section about data compression, he claims the equivalence of data compression and probabilistic modeling as two sides of the same coin and highlights the important role that Bayesian machine-learning models play in the field of data compression (456). The process of optimizing data compression can be viewed as a process of extracting the best model from data, and because of the flexibility provided by Bayesian non-parametric models, some of the best data compression algorithms employs the same idea to achieve better compression rate. Here, a connection is made between machine learning and data compression through the same bridge of probabilistic modeling. Thus from the review of this literature, it is evident that the probabilistic framework can be profound in trying to equate the concepts of machine learning, compression, and search, and is thus an area of interest that we can explore in more details in our project.

Generalization as Search

`http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.121.5764&rep=rep1&type=pdf` [Lorraine]

In ”Generalization as Search,” Tom M. Mitchell claims that the problem of forming a general description of a class of objects given a set of examples and non-examples can be viewed as a search problem. Generalization is cast as a search problem where different methods for generalization would be characterized in terms of different search strategies(203). The generalization language corresponds to an search space of possible solutions, and the learning task is to examine the search space under constraints imposed by training instances, and find generalizations. The paper compares and contrasts multiple strategies, like a general-to-specific partial ordering and several data-directed generalization strategies, which leads to the comparisons of their relative capabilities and computational complexity(224). However, I don’t think those specific strategies apply to the bigger picture of our research topic yet, so I didn’t include their detailed analysis here.

Data Compression Explained

http://mattmahoney.net/dc/dce.html#Section_13 [Rose] [Princewill]

Data Compression Explained is an online book by Matt Mahoney that tries to explain how data compression works. It starts with an information theoretic introduction to data compression and goes on to discuss different codes, models, transforms and lossy compression of images, audio and video. In the information theoretic introduction which we care more about, Mahoney highlights that information theory places hard limits on what can and cannot be compressed losslessly, and by how much:

1. There is no such thing as a "universal" compression algorithm that is guaranteed to compress any input, or even any input above a certain size. In particular, it is not possible to compress random data or compress recursively.
2. Given a model (probability distribution) of your input data, the best you can do is code symbols with probability p using $\log_2 \frac{1}{p}$ bits. Efficient and optimal codes are known.
3. Data has a universal but uncomputable probability distribution. Specifically, any string x has probability (about) $2^{-|M|}$ where M is the shortest possible description of x, and $|M|$ is the length of M in bits, almost independent of the language in which M is written. However there is no general procedure for finding M or even estimating $|M|$ in any language. There is no algorithm that tests for randomness or tells you whether a string can be compressed any further.

Mahoney also explains that modeling is not computable citing work by Solomonoff (1960, 1964), Kolmogorov (1965), and Chaitin (1966) who independently proposed a universal a-priori probability distribution over strings based on their minimum description length. The algorithmic probability of a string x is defined as the fraction of random programs in some language L that output x, where each program M is weighted by $2^{-|M|}$ and $|M|$ is the length of M in bits. This probability is dominated by the shortest such program. We call this length the Kolmogorov complexity $KL(x)$ of x. The best compression we can achieve for any string x is to encode it as the shortest program M in some language L that outputs x.

Mahoney also discusses how compression is an artificial intelligence problem by explaining how prediction or compression could be used to test or measure understanding.

"Minimum Description Length https://www.cs.cmu.edu/~aarti/Class/10704/lec13-MDL.pdf [Princewill] The minimum description length (MDL) criteria in machine learning says that the best description of the data is given by the model which compresses it the best. This is because learning a model for the data or predicting it is about capturing the regularities in the data and any regularity in the data can be used to compress it. Thus, the more we can compress a data, the more we have learnt about it and the better we can predict it. The ideal version of MDL, given by the Kolmogorov Complexity, which is defined as the length of the shortest computer program that prints the sequence of observed data and halts, is uncomputable. Practical versions of MDL are either based on one stage universal codes (known as refined MDL) or two-stage codes (known as crude MDL). The refined MDL suggest that, for a given class of models, pick the model which minimizes the worst case redundancy on the data. This leads to precisely the universal models such as mixture models and normalized maximum likelihood (NML) model. The crude MDL or two-stage code approach is based on the notion that we can specify the descriptive properties of a model for data in two stages - (i) encode the model with some codelength $L(q)$, (ii) encode the data using the model with codelength $L_q(x^n)$. Now pick the model which minimizes the total codelength of the two-stage code: $q_\gamma(x^n) = \arg\min_{q \in Q_\gamma} \{L(q) + Lq(x^n)\}$ The MDL principle can also be used for classification or regression. The two-stage MDL procedure for selecting the best model in a given class would be the regularized least squares framework.

No Free Lunch For Optimization"

`https://ti.arc.nasa.gov/m/profile/dhw/papers/78.pdf` [Santi] [Jacob]

The No Free Lunch Theorems for Optimization paper establishes that under an oracle model of search, two algorithms that use the same number of oracle reveals have the same expected performance if all cost functions have the same likelihood of being chosen. Wolpert and Macready proves this by showing that any algorithm reveals the same sequence of cost on average regardless of what has been previously revealed to the algorithm. The formal proof is done by induction using a statistical model, but it could have been done without statistical model too. The paper also goes on to prove the NFL result for time-dependent cost functions and relate the result to other topics such as information theory and geometry. I have not looked into these topics yet as it seems to be more complex.

The search model used in the paper includes simulated annealing and evolutionary algorithm (where each species is independently evaluated). It does not include techniques like branch and bound or coevolutionary algorithm (where species are evaluated against each other). The authors also went on to prove that there is free lunch for coevolutionary algorithms, that is, some algorithm performs objectively better compared to others.

The authors also mention that they had proved a similar result for statistical inference (machine learning) which I have not explored. I think I will need to spend some time to understand the proof for optimization better before delving into the other one.