No-Free-Lunch and Bayesian Optimality

James L. Carroll, Kevin D. Seppi
3350 TMCB Brigham Young University, Provo UT 84602, USA
james@jlcarroll.net, and kseppi@cs.byu.edu

Abstract

We take a Bayesian approach to the issues of bias, meta bias, transfer, overfit, and No-Free-Lunch in the context of supervised learning. If we accept certain relationships between the function class, on training set data, and off training set data, then a graphical model can be created that represents the supervised learning problem. This graphical model dictates a specific algorithm which will be the "optimal" approach to learning the parameters of any given function representation given the variable relationships. Thus, there is an optimal technique for supervised learning. We reconcile this idea of an optimal technique with the ideas of No-Free-Lunch and show how these ideas relate to the concepts of meta and transfer learning through hierarchical versions of the graphical model.

1 Introduction

One of the goals of meta learning is to automatically determine the best algorithm in a given situation. Thus, the field of meta learning is intimately concerned with the ideas of bias, meta bias, and transfer learning. Such concerns can be conveniently addressed in a Bayesian framework. Such a framework is uniquely suited to represent biases and meta biases through its priors and hierarchical structures [1]. Such a framework is also appealing since it has been shown to produce a certain set of theoretical optimality guarantees.

At first glance these optimality proofs seem to contradict No-Free-Lunch. The No-Free-Lunch theorems are a set of proofs about the nature of generalization with a specific cost function and under the condition that all functions are equally likely. These proofs have been interpreted by some as implying that a bias is required for machine learning and that no one algorithm is better than another over the space of all functions. We will show that the Bayesian approach provides another perspective on the need for a bias in machine learning. Further, we will harmonize the results of the No-Free-Lunch theorems with the results of the Bayesian Optimality theorems and show that, from a decision theoretic perspective, there is a "best" algorithm even in the situation where all functions are equally likely. We will define "best" in terms of decision theory and we will show that this assertion is not at variance with the results of the No-Free-Lunch theorems despite how they have often been interpreted. This can be done because No-Free-Lunch and decision theory each use a different definition of "best."

The remainder of the paper is organized as follows: In section 2 we will propose a flexible graphical model that represents the relationships between the various parts of the supervised learning problem. We will interpret that model in terms of decision theory, and thereby define what we mean by a "best" algorithm. In section 3 we will show how this model compares to traditional techniques. We will show that traditional techniques can be interpreted as approximating the results of the graphical model, and that the model often can explain many of the shortcomings of such techniques, including their tendency to overfit. Furthermore, the model suggests several of the techniques currently used to overcome such shortcomings. In section 4 we discuss the No-Free-Lunch (NFL) theorems in greater detail and propose a Bayesian restatement of the NFL theorems. We also show how the graphical model presents another perspective on the need for a bias in machine learning. In section 5 we then reconcile the ideas of NFL and Bayesian Optimality concluding that a best algorithm actually does exist in the Bayesian context without contradicting the NFL results. In section 6 we will relate these results to meta learning and to transfer. Although there is an algorithm that is best when all functions are equally likely, it is possible to do much better if we can assume that some functions are more likely than others. We call these more likely functions the set of "interesting" functions. Meta learning and transfer learning are concerned with learning this set of interesting functions. We will analyze the task of determining the set of interesting functions from a hierarchical Bayesian perspective. In section 7 we will summarize the observations made by the paper and propose some directions that we hope machine learning research will take as a result of these observations.

2 The Unified Bayesian Decision Theoretic Model (UBDTM)

The supervised classification problem consists of feature vectors **x** that are mapped to an output class value y. A list of **x**, y pairs constitutes a training set. Similarly a test set can be represented by a list of **x**′, y' pairs. Usually there is also an unknown function f that either maps feature vectors to output vectors in the deterministic case $f : \mathbf{x} \rightarrow y$, or from feature vectors to a distribution over output values in the stochastic case $f : \mathbf{x} \rightarrow p(y)$. This formalism is similar, but not identical, to Wolpert's EBF formalism for supervised learning [12].

Without being specific about the distributions involved, we propose that the above elements of a supervised classification problem ($\mathbf{x}, y, \mathbf{x}', y'$ and f) can be thought of as having been drawn from random variables [5]. Of special interest is that we are treating the unknown function itself as a random variable, and this will have significant implications in how the model is used.

Relationships between random variables can be intuitively expressed using graphical models. Nodes in a graphical model represent random variables and edges represent conditional dependencies. Shaded nodes represent observed quantities while non-shaded values represent unobserved quantities. Several different relationships between the random variables in a supervised learning problem are possible, but perhaps the simplest would be the relationship represented by the graphical model in figure 1. This set of relationships is very intuitive, and implies that function outputs y and y' are dependent on the feature vectors **x** and **x**′ and the function f that maps them. Note that this model of variable relationships considers **x** and f to be conditionally independent and that **x**′ may be observed or unobserved depending on the situation. This is not always the case. Unsupervised and semi-supervised learning are all based on the concept that there is sometimes a direct relationship between **x** and f which this model does not capture. The training and test sets are separated because they will be treated differently during learning.

Supervised learning can be seen as the problem of determining $p(y'|\mathbf{x}', \mathbf{x}, y)$, that is, of determining the probability of a class in the test set given its features and the training data. If we accept the relationships among the random variables given in the graphical model of figure 1, then the rules of probability provide an optimal technique for classification. Formally:

$$p(y'|\mathbf{x}', \mathbf{x}, y) = \int p(y', f|\mathbf{x}', \mathbf{x}, y)df \, ,$$

which can be expanded as follows:

$$p(y'|\mathbf{x}', \mathbf{x}, y) = \int p(y'|\mathbf{x}', \mathbf{x}, y, f)p(f|\mathbf{x}', \mathbf{x}, y)df \, ,$$

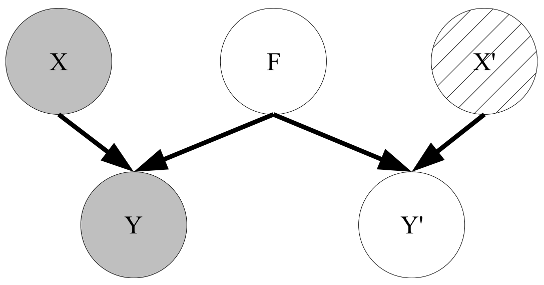

Figure 1: A simple graphical model supervised learning. X and Y represent the training data, while X' and Y' represent the test data. F represents the unknown function that we are trying to learn.

and then, using the independence assumptions of the model we get:

$$p(y'|\mathbf{x}', \mathbf{x}, y) = \int p(y'|\mathbf{x}', f)p(f|\mathbf{x}, y)df \, . \qquad (1)$$

Using Bayes law and the independence assumptions of the model it can also be shown that:

$$p(f|\mathbf{x}, y) = \frac{p(y|\mathbf{x}, f)p(f)}{\int p(y|\mathbf{x}, f)p(f)df} \, . \qquad (2)$$

Equations 1 and 2 form the backbone of our Bayesian perspective on supervised learning. This formulation is surprisingly versatile. Since $p(\mathbf{y}|\mathbf{x}, f)$ and $p(\mathbf{y}'|\mathbf{x}', f)$ are determined by the mapping inherent in f, to create a learning system with a desired bias the user only has to specify $p(f)$ and the rules of probability dictate the rest. When developing a supervised learning system we are no longer interested in selecting the correct algorithm as the algorithm is fixed by the rules of probability. Rather, the designer's task is to select the correct prior (and therefore the correct bias).

It is possible to select $p(f)$ so that we match the representational bias of a wide variety of classical learning techniques. Usually there will be two parts to any specification of $p(f)$: the representation chosen and the parameters for that representation. Let θ represent the parameters of f. For example, $p(f)$ could be distributed so as to have zero mass on all functions that are not composed of the sum of a set of weights, organized into an artificial neural network. The representation of f is now a neural network, and the parameters for this representation θ are the values for the network weights. To complete the specification of $p(f)$, a set of priors should be chosen for the weights. Then equation 2 will determine the true posterior probability distribution over the set of network weights given some training data. Thus, by selecting $p(f)$ according to the structure of an ANN the graphical model provides an optimal technique for learning those weights. A similar approach to learning weights in an ANN was taken by Freitas et. al. [6]. In fact, almost any machine learning algorithm can be recast in these terms so long as it is possible to compute the likelihood of the data given the parameters of the model. This approach has been taken with several learning models including neural networks [6], support vector machines [2], and evolutionary computation [10]. In all of these cases this approach has lead to improved performance over many test cases. This approach is especially useful in evolutionary techniques where the same basic model leads to a technique for finding the value of information which can be used to produce optimal sample locations [11].

Above we have claimed that the rules of Bayesian inference lead to optimal machine learning algorithms but what is the performance measure for which they are optimal? The answer is in decision theory. In practice, classification is not performed for its own sake. In general we do not classify things for the intrinsic value of putting things in classes, rather, we classify things to aid in decision making, and there is some decision that will be made based upon the output of the learning algorithm. As shown in figure 2, the entire decision process can be represented as a decision network tied to the graphical model. Y' is the node most commonly tied to decisions. In the simplest case, there is some outcome O that will depend upon the decision D and the value of Y' (see figure 2). Then the utility U is determined from the outcome O. Many more complex examples could be envisioned, for example, the output could depend on many more things that just Y', however, we will only focus on this simple case.

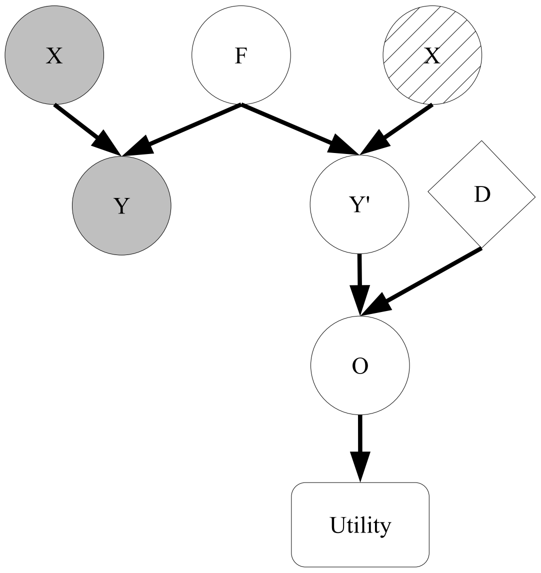

Figure 2: Complete decision theoretic model.

If the posterior distributions for $y'|\mathbf{x}', y, \mathbf{x}$ were computed according to the laws of Bayesian statistics, then the optimal decision $d \in D$ that will maximize the expected utility can be found as follows:

$$\underset{d \in D}{\operatorname{argmax}} \sum_{y' \in Y} U(o) p(o|y', d) p(y'|x', \mathbf{x}, y) \,. \tag{3}$$

We call this the Unified Bayesian Decision Theoretic Model or UBDTM.

The Bayesian optimality proofs show that if the problems you encounter are actually distributed according to your prior, if you compute the posteriors for your parameters following the rules of Bayesian statistics, and if you make expected utility decisions using your posteriors, then you are guaranteed to have an expected utility greater than or equal to decisions made using parameters chosen by any other technique.

Notice that in order to effectively use a classification technique to make general decisions that technique must provide a full distribution over $p(y'|x', \mathbf{x}, \mathbf{y})$. Techniques that simply report the most probable class are not useful for making optimal decisions from a maximum expected utility perspective. For example, if my machine learning algorithm classifies a given mushroom as edible and not poisonous should I eat the mushroom? That depends on the probability of various outcomes if it is poisonous and on how *sure* the algorithm is that the mushroom is not poisonous. Furthermore, it should be pointed out that even in the deterministic case where f maps a feature vector **x** to a single deterministic y, our uncertainty about f should still lead to a probabilistic output over y. We will discuss this issue in greater detail in the next section.

3 Traditional Techniques and Overfit

We will now compare learning the parameters of a given representation for f using the graphical model with traditional techniques. Neural networks provide an illustrative example. Traditional neural networks perform gradient descent on the error surface. If we assume that the output of the neural network represents an approximation to the function's mean, and that error is distributed normally, then finding weights that lower the mean squared error on the data will also raise the likelihood of the training data given the weights $p(y|x, weights)$. Thus gradient descent techniques such as backpropagation are equivalently searching for the $\hat{f}$ that maximizes $p(\mathbf{y}'|\mathbf{x}', \hat{f})$. In other words, backpropagation and other gradient descent training techniques search the space of functions that are formed by the sum of a set of weights for the weight values that maximize the likelihood of the data. This causes these techniques to be plagued by overfit.

One common explanation for overfit is that the hypothesis space was too complex for the amount of training data

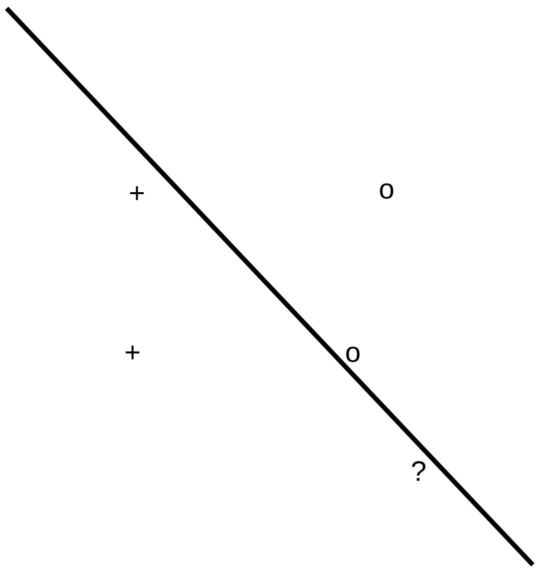

Figure 3: A single hypothesis that maximizes the likelihood of the data.

available. Therefore, one common technique for dealing with overfit involves forcing the hypothesis space to be small for small amounts of data, and then allowing the hypothesis space to expand as the amount of training data grows [7] [3] [8]. However, it can be unclear which hypothesis should be removed to simplify the space, and it can be unclear how much data should be collected before reintroducing these hypotheses. As we shall see, Bayesian techniques provide a formal framework for doing just this. The prior and the likelihood functions together specify how much data would be required before accepting the more unlikely hypotheses. However, the problem of overfit is not just caused by the complexity or simplicity of the hypothesis space searched, but by the fact that the learner was forced to choose a single hypothesis from that space while ignoring its uncertainty that the hypothesis was correct.

By ignoring the learner's uncertainty concerning the hypothesis the learner is forced to make a decision prematurely. This situation is illustrated in figure 3. The dividing line represents a simple hypothesis that maximized the likelihood of the data. If this hypothesis is accepted the unknown location ? must be classified as a +. However, there are many other hypothesis that also maximize the likelihood of the data but which classify the unknown location differently. Equation 1 implies that we should be integrating over all the possible functions and not ignoring them. In the above example this allows us to examine other possible decision boundaries and account for our uncertainty as to which are correct when computing the probability of the class.

By integrating over many different hypotheses, uncertainty will be generated in the probability of the class outputs. This uncertainty will increase as the size of the training set decreases and as the number of functions integrated over increases (as the size of the hypothesis space grows). Inasmuch as overfit is caused by choosing a single hypothesis, equation 1 allows us to reduce overfit, but at the cost of creating additional uncertainty in the output class. Interestingly, this approach of considering multiple hypothesis functions is related to common ensemble techniques.

Typically such techniques generate an ensemble where each element of the ensemble proposes a different approximation for f. Diversification is obtained either by varying the samples via the bootstrap (as in bagging [4]) or by using several different learning algorithms with the same set of samples. The ensembles then vote for the output class. Traditional techniques stop here, however, the proportions of votes for each class could be interpreted as a posterior probability distribution. If this approach is taken, such techniques can be thought of as approximating the integral in equation 1 for a few sample hypotheses. Unsurprisingly, this approach can greatly reduce overfit.

4 No-Free-Lunch and Bias

By integrating over the possible functions we have traded overfit for uncertainty. However, the question remains: how uncertain should we be in our classification of a feature vector that is not in our training set. Examples that are not in the training set are often called "off training set" examples. Wolpert and Mitchell both showed that in the discrete case there are as many hypotheses that classify a given off training set feature one way as that classify it another [9] [12]. They proved this result in the discrete case.

One reasonable probabilistic representation of this situation would be to put a Multinomial distribution over the discrete space of all possible functions. Unlike the ANN example above, a Multinomial distribution allows us to put some probability on all finite discrete functions, and it allows us to select parameters such that this probability is equal for all these functions (what we will call the uniform prior case):

$$p(f) \sim Multinomial(1, \boldsymbol{\theta}) \,, \tag{4}$$

where

$$\theta_i = p(f_i) = 1/j \,,$$

and where j is the number of unique functions produced by our discretization. The No-Free-Lunch proofs dictate that under that assumption, equation 1 will always return a uniform posterior distribution over those $p(y'|x', \mathbf{x}, y)$ where x' is not found in the training set. The proof of this conjecture is beyond the scope of the current work, but it flows directly from Mitchell and Wolpert's existing No-Free-Lunch proofs

[9] [12]. Intuitively this is easiest to see in the deterministic case although it is true for non-deterministic cases as well. In the deterministic case, we can throw out any hypothesis that does not match the training set. Formally:

$$p(y|\mathbf{x}, f_i) = \begin{cases} 1 & \text{if } y = f_i(\mathbf{x}) \\ 0 & \text{otherwise} \end{cases}$$

it can be simply shown that given the prior in 4, and the above likelihood function, then the posterior probability of the functions given the data is:

$$p(f_i|\mathbf{x}, y) = \begin{cases} 1/c & \text{if } y = f_i(\mathbf{x}) \\ 0 & \text{otherwise} \end{cases}$$

where c is the number of functions that are consistent with y and $\mathbf{x}$. Thus, a uniform prior over functions leads to a uniform posterior over those functions that are consistent with the training data. Mitchell proved that there will be as many remaining consistent functions that classify off training set values one way as there are that classify them another way. Since all of these remaining functions are equally likely, the posterior probability for $y'|\mathbf{x}', \mathbf{x}, y$ when $\mathbf{x}'$ is off training set is also uniform.

A Bayesian restatement of the No-Free-Lunch proofs would be that *a uniform prior for f will yield a uniform posterior for $y'|\mathbf{x}, y$ when $\mathbf{x}'$ is off training set.* If a more informative posterior is desired a more informative prior is necessary. Thus, the our graphical model leads to another way of looking at the No-Free-Lunch proofs.

The UBDTM can help with more than just providing an intuitive interpretation of the No-Free-Lunch proofs. No-Free-Lunch and Mitchell's related paper are often interpreted as showing that a bias is necessary for machine learning. For example, Mitchell titles his paper "The Need for Bias in Machine Learning" and then essentially proves that a uniform prior leads to a uniform posterior. Wolpert's No-Free-Lunch papers essentially do the same thing. If we accept as axiomatic that a uniform prior is "unbiased" and that a uniform posterior is a failure, then these papers do indeed show that unbiased learning is impossible.

However, neither of these statements are axiomatic from a Bayesian perspective. Of course, we still believe that a bias is necessary for generalization, but for a very different and more fundamental reason. For a Bayesian the so called "unbiased case" is actually a very specific prior, namely the uniform prior which *is* a prior *and* a bias. Furthermore, the uniform posterior is far from a failure, it is a perfectly acceptable and even potentially useful answer for making decisions that will maximize expected utility. Thus, from a Bayesian perspective Mitchell and Wolpert's proofs showed that one specific prior leads to one specific posterior, not that a bias is necessary for generalization.

From the perspective of the graphical model, the need for a bias is clear: Bayesian inference is impossible without a prior, and therefore without a bias. If we removed the bias and attempt to use Frequentist inference we fare no better since there will either be no data for off training set examples (and therefore the maximum likelihood estimator is undefined) or else information from one part of the function (the on training set part) must influence our belief about off training set locations (which would also constitute a bias). Thus, from a Bayesian perspective, a bias is necessary in machine learning, but not for the reasons usually given.

Another advantage of thinking about supervised learning from this perspective is that the bias of any system is explicit. We know that traditional techniques have a bias, however it can often be difficult to determine what that bias is. In the graphical model the bias is explicit in $p(f)$. The representation chosen for $p(f)$ captures the representational bias of the learning system, while the priors chosen for $p(f)$ represent the preferential bias of the algorithm.

5 No-Free-Lunch and Optimality

The No-Free-Lunch theorems have been interpreted as saying that all machine learning algorithms are approximately equal, and that no one algorithm can outperform another, at least over the uniform space of all functions. On the other hand, Bayesian techniques are often said to be "optimal" given a specific prior. However, we have claimed that the uniform prior over all functions is just a prior like any other, so the Bayesian approach should be optimal, even in this so called *a priori* case. Yet this is the same case for which No-Free-Lunch claims that all algorithms perform equally well. How is this reconciled with No-Free-Lunch?

First, it will be necessary to remember what we mean when we say that Bayesian techniques are optimal. Bayesian techniques are optimal with respect to a specific metric, namely expected utility. No-Free-Lunch was proved for a very specific cost function, namely misclassification error rate. For cost functions other than misclassification error rate it is possible to have *a priori* distinctions between learning algorithms. Wolpert wrote: "if the error function induces a geometrical structure over Y then we can have *a priori* distinctions between learning algorithms." Although misclassification error rate can be represented as a specific utility function which does not impose such a geometric structure, the broader class of utility functions can impose such a structure and are therefore outside of the realm in which No-Free-Lunch holds.

The proof is by example. Several examples of such utility functions could be provided. We selected a simple classification task, where the goal is to classify a feature as either a or b. The utility function is given in table 1. Note that the utility function depends on a variable we called g. By varying g we were able to create different utility functions each with different expected utility off training set.

Table 1: Simple utility function that can lead to *a-priori* distinctions between learning algorithms.

	class=a	class=b
Action=a	g	$1-g$
Action=b	0	1

We analyzed the relative performance of the Bayesian approach as compared to random for various choices of g. These results can be seen in Figure 4.

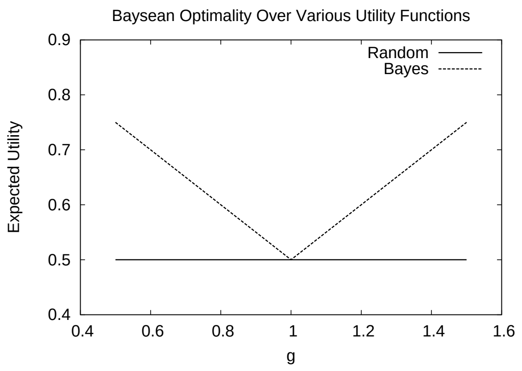

Figure 4: Expected utilities of making so called *a priori* off training set decisions either randomly or according to the laws of Bayesian statistics and decision theory. The utility function varies in the x axis, and when it is equivalent to misclassification error rate (g=1) there is no-free-lunch.

Notice that for most of the possible utility functions the technique based upon the Bayesian algorithm's uniform posterior outperforms the random decision (and therefore any decision based upon random distributions for $p(y')$). In this case there is an optimal output from the classifier. The optimal decision was made using the uniform distribution returned by the rules of Bayesian statistics. This means that if the classifier returned a distribution other than uniform the algorithm would have been suboptimal for some utility functions. This illustrates that some simple utility functions do indeed create *a priori* distinction between learning algorithms. Note also that at the point where $g = 1$ the utility function becomes equivalent to misclassification error rate. At this one point, as predicted by NFL, the Bayesian algorithm performs no better or worse than random. When the cost metric becomes misclassification error rate all algorithms perform equally well, and therefore no better than random. However, for most other utility functions the optimal Bayesian algorithm performs better than random, and will perform better than or equal to any other technique that could be employed.

Furthermore, these utility functions other than misclassification error rate are quite common in practice. It is more often the case that one type of error is more damaging than another. Furthermore, it is often the case that precision is more important than recall or visa versa. Thus when using classification algorithms to aid in decision making it is only in rare cases that the error function actually matches misclassification error rate where No-Free-Lunch holds. In an academic sense it has been convenient to think about the performance of classification systems in terms of misclassification error, however, in the real world the utility of a failed classification is rarely exactly opposite the utility of a success.

Thus, there is an optimal off training set distribution over classes even given the uniform prior, and that is the uniform posterior. For decision making it is just as important to express uncertainty as it is to express what is known. Even the uniform posterior contains important information for making decisions. This also means that there is an optimal classifier for off training set examples even over the uniform space of all functions, namely any classifier that returns the uniform distribution (as Bayes does if a Multinomial uniform prior is used). However, a Bayesian algorithm will also be optimal with respect to expected utility for any prior over functions. In UBDTM such biases are easy to create, precise, explicit, and transparently represented in the user's choices for $p(f)$. Determining what this prior should be, or placing a distribution over possible priors is the goal of meta learning.

6 Meta Learning and Transfer

There is a best algorithm for producing $p(y'|\mathbf{x}', \mathbf{x}, y)$ for a specific prior over $p(f)$, however, if the prior for $p(f)$ is no more expressive than the uniform prior, then the posterior will also be uniform. Although this posterior can provide some useful information for decision making, a more informative prior would be preferred.

How should a more informed prior $p(f)$ be chosen and what justification could be used for a distribution other than uniform? It is often assumed that the universe provides problems that we might want to solve according to some distribution other than uniform, $p(m)$. If we could characterize this distribution of functions then we could create much more informed priors than the uniform prior.

If meta learning is recast in terms of a Bayesian graphical model, then the task of meta learning is to attempt to learn the distribution over interesting functions which we will designate $p(m)$. The available data for learning this function would be a set of learned functions encountered in real life situations $p(f)_i$. Thus we would want to compute $p(m|f_1...f_n)$. Furthermore, the distribution over m would make an effective prior for learning a new function $p(f)_{n+1}$. Thus meta learning and transfer learning can both be expressed as a hierarchical version of UBDTM [1].

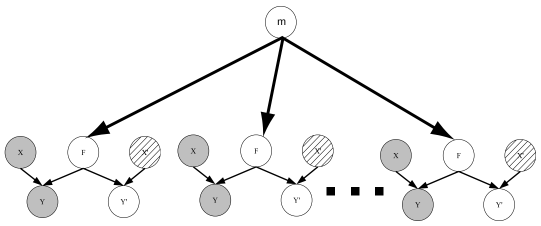

Figure 5: Hierarchical Bayesian model for meta learning.

This more complex model requires the specification of a prior over the meta distribution $p(m)$ and the specification of $p(f|m)$. The distribution $p(m|f_1...f_n)$ would be determined using Bayes law. Now the probability of class y'_{n+1} for an off training set feature $\mathbf{x}'_{n+1}$ in a new function f_{n+1} is given by:

$$\begin{aligned} & p(y'_{n+1}|\mathbf{x}'_{n+1}, \mathbf{x}_{n+1}, y_{n+1}, f_1...f_n) = \\ & \iint p(y'_{n+1}|\mathbf{x}'_{n+1}, f_{n+1}) \\ & p(f_{n+1}|\mathbf{x}_{n+1}, y_{n+1}, m) \\ & p(m|f_1...f_n) df_{n+1} dm \, . \end{aligned} \tag{5}$$

Of course this does not completely absolve us from the need for an informed prior since if we begin with a uniform prior over $p(m)$, then we will end up with a $p(f)_{n+1}$ which is uniform for all locations not in the training sets of the other tasks. This will in turn will lead to a uniform distribution over off training set function outputs y'_{n+1}. Our meta learner needs a bias just as much as a standard learner did. Again this is not surprising since Wolpert addressed this meta learning issue when he discussed unknown distributions other than uniform in [12, see footnote 4].

7 Conclusions

We have examined the connections between overfit, uncertainty, and NFL, showing that as more possible functions are considered, off-training-set uncertainty increases until the posterior becomes uniform once all functions are considered equally likely. We have shown that from a Bayesian perspective the No-Free-Lunch theorems can be restated as *a uniform prior over functions lead to a uniform off training set posterior over classes.* We would propose this as a Bayesian restatement of the NFL theorems for Base learning. We have also shown that there is a best supervised learning algorithm from a decision theoretic perspective, even over the uniform prior case, and reconciled this result with the NFL theorems based upon the different cost functions involved.

We are now in a position to answer the question "Is there an NFL Theorem for Meta-learning?" We have shown that meta learning can be seen as a variation of hierarchical Bayesian learning in the UBDTM. If the variable relationships of the UBDTM are accepted then it is clear that Bayesian inference cannot be performed without a prior $p(m)$. Therefore a prior (and thus a bias) is just as necessary for meta learning as it was for base learning. Furthermore, it is still the case that: *a uniform prior over the meta parameters leads to a uniform off training set posterior over classes.* We would propose this as a Bayesian restatement of the NFL theorem for Meta-learning.

Many of the results and observations of this paper are already known by some in the machine learning community, however, the techniques, implications, and conclusions that can be drawn from them are not universally understood and are under applied. It is our hope that these ideas will lead to:

1. an increased focus on algorithms that return a distribution over classes rather than simply reporting the most likely class so that these results can be more useful in decision making

2. an increased use of utility in actual applications as a metric for measuring the success of supervised learning systems

3. an increased awareness of the learner's uncertainty over possible functions, including functions other than the one that maximizes the likelihood of the data, and in attempts to account for that uncertainty in order to avoid overfit

4. an increased interest in learning the parameters of various models in a Bayesian way

5. further experimentation with hierarchical Bayesian models for meta learning and transfer learning

6. an increase in heuristics based upon and justified through their approximation to the Bayesian solution.

Many Bayesian techniques can be computationally intense, however, an increased understanding of the theoretically optimal solution can often lead to better heuristics, and an increase in understanding of why such heuristics work. For example, bagging and related techniques can be interpreted from a Bayesian perspective, as a Heuristic for solving the more correct yet often intractable integral.

References

[1] J. Baxter. A bayesian/information theoretic model of bias learning. *Proceedings of the ninth annual conference on Computational learning theory*, pages 77 – 88, 1996.

[2] C. M. Bishop and M. E. Tippling. Bayesian regression and classification. *Advances in Learning Theory: Methods, Models and Applications*, 190:267–285, 2003.

[3] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. Warmuth. Learnability and the vapnik-chervonenkis dimension. *Journal of the Association for Computing Machinery*, 1989:929–965, 1989.

[4] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[5] W. Buntine. *A Theory of Learning Classification Rules*. PhD thesis, Sydney, February 1990.

[6] J. de Freitas, M. Niranjan, A. Gee, and A. Doucet. Sequential monte carlo methods for optimisation of neural network models, 1998.

[7] J. Halbert L. White. Learning in artificial neural networks: A statistical perspective. *Neural Computation*, 1:425–464, 1989.

[8] D. Haussler. Decision theoretic generalizations of the pac model for neural net and other learning applications. *Information and Computation*, 100(1):78–150, 1992.

[9] T. M. Mitchell. The need for biases in learning generalizations. Technical Report CBM-TR-117, New Brunswick, New Jersey, 1980.

[10] C. K. Monson. *No Free Lunch, Bayesian Inference, and Utility: A Decision-Theoretic Approach to Optimization*. PhD thesis, Brigham Young University, Department of Computer Science, 2006.

[11] C. K. Monson, K. D. Seppi, and J. L. Carroll. A utile function optimizer. In *The Proceedings of the IEEE Congress on Evolutionary Computation (CEC) (accepted)*. IEEE Press, Sept 2007.

[12] D. H. Wolpert. The supervised learning no-free-lunch theorems. Technical Report 269-1, NASA Ames Research Center, 2001.