Planning Document

John Liddell

jl0000285@gmail.com

1 Background and Significance

Determining what algorithm to use when analyzing a dataset is a problem as old as machine learning itself. The primary issue with regards to choice of learning algorithms lies with the fact that learners tend to have differing performances depending on the sort of data that the learner is learning on. The decision of what algorithm is most likely to perform well with some given dataset is often times left up to an expert, an individual that is well versed in machine learning literature and who understands the problem well enough to make an informed decision. Often times such an expert is not available, however, in which case the fallback is often times the application of a meta-learner which is a learning algorithm that produces a learning machine that it believes will best do the job of fitting a given dataset when given that dataset.

The issue of what learning algorithm performs best on what kinds of dataset can be summerized by noting that the elevated peformance of any optimization algorithm for one class of problems will be payed for by reduced performance of said optimization algorithm on other classes of algorithms. [1] In otherwords, no one optimization algorithm or even strategy should simply be better than any other, it should depend on the class of problem for which one wishes to find an optimum solution.

Metalearning machines hope to discover this best algorithm with an algorithm. As such, they themselves should be subject to this constraint: no one metalearning strategy should simply be better at determining what learning algorithm will best perform on a given dataset; the choice of which algorithm best determines the best algorithm should itself be succeptable to this "No free lunch" constraint.

2 Problem Statement and Hypothesis

The goal of this study is to determine whether or not the "No free lunch" optimization problem constraint applies to metalearning systems or whether

or not some metalearning stratagies are strictly better than others. Being that metalearning machines are just a type of machine learning algorithm leads me to presume the null hypothesis to be yes, metalearning machines are subject to the "No free lunch" theorem. I also do not currently believe they should be free from the NFL theorem and am thus adopting the null hypothesis as my own.

3 Summary of theories and algorithms

In order to test the validity of the NFL theorem as it applies to metalearning systems, we have need of metalearning strategies to compare and producable learning machines. The metalearning strategies that will be subject for comparison are "Active Metalearning" [3], the sampling strategy outlined in "Predicting relative performance of classifiers from samples" [4], and a generic bruite force approach in which every algorithm within a given metabase will be tested. A summary of each of the relative algorithms and theories needed to understand the work now follows.

3.1 No Free lunch theorem

Introduced in Wolpert and MacReady's "No Free Lunch Theorems for Optimization" 1997 paper, the No Free Lunch theorem states that the perfomance of all algorithms when averaged out across all datasets should be the same, that is to say there is no one algorithm that is universally the best. The root cause of this observation is in that differing algorithms make different assumptions about the distributions from which the data the algorithms work with arises. A learning algorithm with an implicit assumption of a random distriubtion will have a far lower test case classification accuracy than an algorithm that assumes a gaussian distribution if the distribution from which the set of observed samples derives from is truly normal and vice versa, if the distribution is truly random the gaussian classifier's accuracy will suffer relative to the classifier with a random assumption.

3.2 K-Means clustering

The K-means algorithm plays a core role in the system that will perform the experiment, being as how it is both the algorithm that will be used to compare new datasets to the metabase during evaluation and as how it is also one of the machines that can be produced by the metalearners. As such it is the first of the producable machines that I will describe.

The objective of the k-means algorithm (which results in a k-means model) is to partition a dataset into k groups such that the points within some group are all closest to the mean of that group than they are to any other group. A clear informal explanation of the work that the k-means

algorith performs was given by James McQueen in 1967: "...the k-means procedure consists of simply starting with k groups each of which consists of a single random point, and thereafter adding each new point to the group whose mean the new point is nearest. After a point is added to a group, the mean of that group is adjusted in order to take account of the new point. Thus at each stage the k-means are, in fact, the means of the groups they represent."[2] Formally stated, given an integer k and a set of n data points in $\mathbb{R}^d$ the K-means algorithm seeks to minimize Φ, the over all total summed in class distance between each point and its closest center such that

$$\Phi = \sum_{x \in X} min_{c \in C}\ x - c^2$$

[8].

The k-means model is a type of gaussian mixture model that is trained with a procedure called expectation maximization. Given a set of distributions with missing data , mixture models tend to have derivitaves that are either difficult to define or entirely undefinable. On the other hand, the calculation of some ML/MAP estimates for some set of models can generally be calculated with little difficulty if every point within the distributions is known (at which point our learner would obviously have nothing to do) and thus calculus would be entirely unneccessary (i.e it wouldn't matter that the derivitive cant be defined). Expectation maximization uses this fact in order to obtain an estimation of the ML/MAP in a roundabout way. The algorithm consists of two steps. First, an estimate as to what the expected value of the hidden data is based off the current guess for the parameters is made. Then the likelihood function for the parameters is maximized under the assumption that the data discovered in the previous step is complete i.e that there is no longer any hidden data. These steps are then repeated until some convergance criteria is met. The k-means is exactly this type of algorithm but with the covariance matrix $\Sigma_k = \rho^2 * I_D$ and the mixing weights $\pi_k = 1/K$ all being fixed such that the only free parameters are the cluster centers $\mu_k \in \mathbb{R}^D$ and such that the hidden data that is the ground truth label of the data points.

3.3 Linear Regression

Linear regression is one of the most common and oldest machine learning techniques within literature. It asserts that the response is a linear function of the inputs. [7]. This relation takes the following form:

$$y(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + \epsilon = \sum_{j=1}^{D} w_j x_j + \epsilon$$

where $w^T x$ represents the inner or scalar product between the input vector x and the model's weight vector w^T, and ϵ is the residual error between our linear predictions and the true response.

To fit a linear regression model, the least squares approach is usually used. Given some "overdetermined" linear system (that is to say a system in which there are more equations than parameters) one can write an expression for the sum of squares of the system

$$S(\beta) = (y_1 - \beta x_1)^2 + (y_2 - \beta x_2)^2 + ...(y_3 - \beta x_3)$$

then take the partial deriviate of this sum of sqaured deviations with respect to each of the components of β, set them to zero, then solve the resulting equations to directly determine what the values of the parameters are that minimizes the sum of the squared errors of the system.

3.4 Naive Bayes

The Naive Bayes classifier algorithm fits a set of data to Bayes theorem with a strong assumption of feature independance. Given a set of discrete-valued features $x \in 1, ..., K^D$, we can calculate the class conditional density for each feature, then with our assumption of independance easily generate a guess at what the class should be for a new input by multiplying the conditional likelihood values for each of the new inputs features times the prior on the desired to be known class, that is to say

$$p(y|\mathbf{x}) \propto p(y) \sum_{j}^{D} p(x_j|y)$$

. The calculation of the posterior probability for a new example can be done so "by hand" or can be done so from distributions that are infered from the provided data.

3.5 Support Vector Machine

The support vector machine is a two-group classification algorithm that attempts to find a hyper plane that seperates the inputs within the given input space with a maximum margin of seperation between the hyper plane and the "support vectors", those vectors on either side of the hyper plane that are closest too it. The support vector machine can use "kernels", to map input vectors into non-linear high-demensional feature space without actually calculating the position of the vectors within that feature space [5]. The kernel accomplishes this by calculating the distance between (or similarity) of its two input vectors in this space without reference to their exact position within this higher space. This then allows the computation of a linear seperation between the points in this higher dimensional space

which translates into a non-linear seperation for the vectors in their original lower dimensional space where a seperation might other wise not have been discoverable.

3.6 Neural Networks

A neural network is a type of machine learning algorithm that mimics the interconnectivity of animal brains in order to automatically discover rules to classify given inputs. Being that it is one of the most flexible learning algorithms within literature (actually able to approximate any continuous function)[6], its inclusion within a metalearning system is almost mandatory. Genrally, such a system works by first being presented with a set of classified or unclassified inputs. Said neural network system will then attempt to make a decision on these inputs on which an error value will then be assigned. The system will then see some kind of correction function applied to it. This process will continue until the system has exhausted its supply of training data, at which point it will hopefully have discovered a strong set of rules for peforming whatever work it is that it was designed to perform.

The type of neural network that will be used within this thesis is what is called the feed-forward neural network (multilayer perceptron aka MLP). The feed forward neural network is essentially a series of logistic regression models stacked on top of each other, with the final layer being either another logicstic regression or linear regression model depending on whether or not a classification or regression problem is being solved.[7]. The leftmost layer of this stack is called the input layer and consists of a set of neurons

$$x_i|x_1, x_2, x_3..., x_m$$

representing the input's features. Each neuron in the hidden layer transforms the values from the previous layer via weighted linear summation

$$w_1X_1 + w_2x_2 + ...w_mx_m$$

[9] which is then passed into a non linear-action function g(), such as the logistic function or the hyperbolic tangent function. It is important to note that g must be non-linear, otherwise the entire model will collapse into a large linear regression model of the form $y = w^T(V * x)$. [7]

3.7 Brute force metabase

The most basic meta machine learning algorithm involves the collecting of the run statistics of the set of machines that a given meta learner can produce applied to a metabase with a clustering algorithm to produce results with new datasets. This version of metalearning will act as a sort of control for this study, the results of the more complicated meta learning strategies that

follow only really mean anything with respect to how long it would've taken to train and how long it would've performed if we had just trained it via brute force.

3.8 Active Meta Learning

The first of the metalearning strategies implemented within the study; Active Meta Learning is a meta learning technique "that reduces the cost of generating Meta examples by selecting relevant Meta examples" [3]. What this entails is a decision on what datasets to allow into a metalearners metabase, rather than analyze every candidate meta base dataset an active metalearner will analyze the next dataset with the highest uncertainty. The relative uncertainty between two datasets is defined by

$$\delta(V_x, d_i, V_x, d_j) = \frac{|V_x, d_i - V_x, d_j|}{Max_{k \neq i}(V_x, d_k) - Min_{k \neq i}(V_x, d_k)}$$

where V_x, d_k is the value of some metaparameter V_x for dataset d_k, $Max_{k \neq i}(V_x, d_k)$ is the maximum value of V_x, d_k when dataset i is removed and $Min_{k \neq i}(V_x, d_k)$ is its corrosponding minimum.

3.9 Predicting relative performance of classifiers from samples

The second of the metalearning strategies implemented within this study is one in which a representive subsection of the metabase is trained with each algorithm entirely, after which point the rest of the metabase under goes a sort of curve sampling analysis in which the results of future learning is predicted based off what dataset the new datasets learning curve most resembles. [4] As with the other two meta learning strategies, the label for new datsets is the determined via clustering with the datasets within this post trained metabase.

4 Methodology

The form in which this experiment will take is that of a program in which a set of meta-learners will be built, each operating in accordance with one of the meta-learning strategies described in the previous section. The system and each of its components will be coded in Python and run from either a linux bash or windows powershell terminal. What follows is description of the important non-standard library modules from which the system will be built.

4.1 Libraries and hardware

4.1.1 scikit-learn

Scikit-learn is a machine learning library that is written almost entirely in python, with a few bindings in Cython being present to increase performance where applicable. The use of Scikit-learn within this project comes with several significant advantages: the individual machine algorithms will already be written and optimized, the algorithms will be garunteed to work in pretty much any environment, and the relative performance of the algorithms is garunteed, that is to say an algorithm that should theoretically perform better with some given dataset (for example a gaussian svm kernel with data points from a gaussian distribution vs any other kernel) will do so leading to more honest metalearning databases.

4.1.2 sqlalchemy

SQL-Alchemy is an Object Relational Mapper that allow application developers to interface with databases via python. This makes it possible to persist and manipulate the large amount of data associated with an experiment of this type in a clean, consistant, and comprehensible fashion.

4.2 unit hardware

The unit on which the experiment will be carried out on has still not been determined, though in a pinch my personal desktop computer might possibly suffice.

5 Timeline

- Gather datasets, apply learning algorithms to determine which datasets can be used or need to be cleaned
- Build metalearning system
- Set metalearning system up on applicable machine and gather evidence by running system
- Analayze evidence to determine likelihood of hypothesis
- Complete thesis paper
- Defend thesis

References

[1] David H. Wolpert; William G. Macready *No free lunch Theorems for Optimization* IEEE Transactions on evolutionary computation vol 1 no.1 April 1997

[2] McQuenn *Some methods for classification and analysis of multivariate observations*

[3] Bhatt, Thakkar; Bhatt, Ganatra *Ranking of Classifiers based on dataset characterstics using active meta learning*

[4] Leite, Rui; Brazdil, Pavel *Predicting Relative Performance of Classifiers from Samples*

[5] Vapnik, Vladimir; Cortes, Corinna *Support-Vector Networks* Machine Learning, 20, 273-297 1995

[6] Hornik, Kurt *Approximation Capabilities of Multilayer Feedforward Networks*

[7] Kevin P. Murphy *Machine Learning: A probabilistic Perspective*

[8] Arthur, David; Vassilvitskii, Sergei *k-means++: The advantages of careful seeding*

[9] scikit-learn.org *https://scikit-learn.org/stable/modules/neural networks superivsed.html*