On the Structure of Algorithm Spaces

Adam Peterson, Tony Martinez and George Rudolph

Abstract— **Many learning algorithms have been developed to solve various problems. Machine learning practitioners must use their knowledge of the merits of the algorithms they know to decide which to use for each task. This process often raises questions such as: (1) If performance is poor after trying certain algorithms, which should be tried next? (2) Are some learning algorithms the same in terms of actual task classification? (3) Which algorithms are most different from each other? (4) How different? (5) Which algorithms should be tried for a particular problem? This research uses the COD (Classifier Output Difference) distance metric for measuring how similar or different learning algorithms are. The COD quantifies the difference in output behavior between pairs of learning algorithms. We construct a distance matrix from the individual COD values, and use the matrix to show the spectrum of differences among families of learning algorithms. Results show that individual algorithms tend to cluster along family and functional lines. Our focus, however, is on the structure of relationships among algorithm families in the space of algorithms, rather than on individual algorithms. A number of visualizations illustrate these results. The uniform numerical representation of COD data lends itself to human visualization techniques.**

I. Introduction

Several machine learning algorithms and algorithm families have been specified and developed by the machine learning research community. These include algorithms like neural networks, decision trees, instance based learners, rule based systems and belief networks. Each algorithm makes some assumptions about the problems to which it is applied. Each algorithm performs most effectively on problems where those assumptions are more or less satisfied, and poorly on others. These assumptions bias the algorithm towards one solution or another, and this bias is a necessary part of learning [1].

When we choose an algorithm to solve a problem, and the algorithm performs poorly, we then must decide what algorithms to try next. We may want to try an algorithm that is similar, but differs in some respect, or we may want to try a completely different algorithm. When the performance of one or more algorithms is known, we can use this to estimate the performance of other algorithms on our task. This is most convenient if it can be done without having to train and execute these other algorithms first. Performance can be used to help guide learning algorithm selection without having to measure the performance of every algorithm on the target task.

Adam Peterson is with Adobe Systems, Orem UT, 84097, USA (email: adam@axon.cs.byu.edu). Tony Martinez is with the Department of Computer Science, Brigham Young University, Provo, UT, 84602, USA (email: martinez@cs.byu.edu). George Rudolph is with the Department of Mathematics and Computer Science, The Citadel, Charleston, SC, 29409, USA (email: george.rudolph@citadel.edu).

The classical measure of algorithm performance is accuracy. Accuracy measures how well an algorithm generalizes on a task. Two algorithms with similar accuracy may give different answers, however. It is these differences that provide fertile ground for improving algorithm performance, and therefore accuracy alone is an insufficient measure of performance.

Peterson [2] proposed the COD distance metric, which is a method that quantifies the behavioral differences between algorithms. We use COD data to compute and visualize the relationships among learning algorithms as an aid to increasing our understanding of the machine learning problem space and algorithm space. As we discuss COD concepts, we give a method for visualizing regions of disagreement. To explore differences and similarities among algorithms, we selected seventeen machine learning algorithms, representing eight different families, and we chose thirty tasks from the UC Irvine Machine Learning Repository [3]. A distance matrix is constructed from the individual pairwise COD values. When viewed as a dissimilarity matrix, statistical techniques like multidimensional scaling algorithms [4] (MDS) can scale the data to two or three dimensions. This enables us to generate visualizations that encode positional data, like a scatterplot, and analyze the results. A two-dimensional plot is used here because it is convenient for black-and-white printing where the user is not able to interact with, nor filter the data. Three dimensional and interactive visualizations are more productive when using a computer.

These visualizations show that members of many learning algorithm families, such as decision trees, tend to cluster near each other. Other algorithm families, such as support vector machines, group much more loosely. We can also see which learning algorithm families fall near each other. Neural network-based algorithms fall near logistic regression algorithms. On the other hand, instance-based algorithms are farther from Naïve Bayes.

COD can be used to compare individual algorithms, however the focus of this paper is on the structure of the algorithm space–that is, showing the relationships among algorithm families. One feature of the COD method is that the uniform numerical representation lends itself to human visualization techniques, machine meta-learning techniques, to the search for new algorithms where current algorithms perform poorly, and to analyzing theoretical underpinnings like the No Free Lunch Theorem [5]. This paper focuses on the first of these four areas.

The rest of the paper is organized as follows: Section II discusses basic concepts and definitions. Section III overviews related work. Sections IV and V describe the COD method and the tasks and data used. Section VI gives results

978-1-4244-9636-5/11/$26.00 ©2011 IEEE

and analysis. Section VII briefly discusses other factors that may affect similarity. Section VIII is the conclusion.

II. Concepts

A *learning task* is an interesting problem that someone wants to solve. We call the set of interesting problems P_I, to distinguish it from the universe of all possible problems. Problems in P_I exhibit some kind of regularity that we can observe and attempt to model.

A *hypothesis* is an executable model that attempts to solve a task with acceptable accuracy. A *learning algorithm*, or simply *algorithm*, is a procedure that, when given a learning task, produces a *hypothesis*. In this paper, the terms *hypothesis* and *classifier* will be used interchangeably. We use h_0 to denote "the perfect classifier" hypothesis that gives the correct output for every input pattern. The distance metric is called *Classifier Output Difference* because all of the algorithms studied thus far are classifiers. It is straightforward to apply the COD method to other kinds of algorithms, but that is a subject for future work.

When considering how to construct a model of the structure of the algorithm space, a number of approaches could be taken to compare learning algorithms. One approach would be to compare the types of decision surfaces employed by one learning algorithm as opposed to another. A second approach would be to evaluate the capacity for one learning algorithm to simulate, or mimic, another. This would be problematic when dealing with learning algorithms that are capable of shattering, or arbitrarily dichotomizing, any set of consistent patterns and therefore capable of simulating any other hypothesis.

The COD method takes a different approach. First, it provides a way to measure the distance between two hypotheses. The distance for a task is calculated by counting the number of patterns for which two hypotheses give different output values, and dividing by the total number of patterns presented. The result estimates the probability that they disagree on the classification of patterns, or the frequency that the hypotheses disagree with each other. More than comparing aggregated accuracy values, this metric looks at the pairwise output behavior on each instance from the task. The COD distance between two algorithms is the expected distance between the hypotheses produced by the respective algorithm on a task.

The distance between two learning algorithms over a collection of learning tasks is estimated by averaging the distance between the algorithms over each individual task. The data may be aggregated in other ways as well. The distances for each task across all algorithms can be obtained by averaging the values for each algorithm on a given task, for example.

This research attempts to improve our understanding of learning algorithms and where they exhibit qualitative behavior differences. If two learning algorithms behave similarly on learning tasks, their internal differences in the way they compute their answers are less important, since from a functional perspective the algorithms essentially perform the same task.

III. Related Work

The COD measure defined by Peterson in [2] is used in this paper. Kuncheva surveyed other diversity measures [6]. The COD measure is similar to the Disagreement Measure, in the concept learning (two output classes) case, but different in the multiclass case. Some of the surveyed metrics could also be used to measure algorithm similarity at the hypothesis level. The COD metric focuses directly on difference in output behavior, whereas these other measures use accuracy at the pattern level. Some of Kuncheva's metrics also correlate negatively with behavior difference.

Quantifying distances between learning algorithms has tie-ins with *meta-learning*, where machine learning strategies are applied at more than one level in the problem solving process. This idea is used in Pfahringer's *landmarking* [7]. Landmarking works to locate learning tasks in the space of all learning tasks. Problems exist in a space of learning tasks, and their relative positions are estimated by evaluating the success of simple learning algorithms on the task. Landmarking is, in some respects, the inverse approach of COD. In landmarking, learning tasks are measured relative to some simple learning algorithms. COD measures distances between learning algorithms and hypotheses by using their performance on learning tasks.

Zheng [8] has a simpler approach to learning task similarity. This approach uses several simple metrics to categorize learning tasks, such as number of inputs and whether all inputs are real-valued, for example.

Brodley [9] advocates using the different strengths of different algorithms to follow a hybrid approach. Brodley's research categorizes algorithms into "model classes" based on the form. Each model class represents a hypothesis that is used for generalization. It supports automatic model selection by searching through model classes, and also suggests constructing hypotheses with heterogeneous mixes of representations.

Displaying data, particularly data that changes, requires careful planning. Visualization is commonly described as a seven-stage process [10]: Acquire, parse, filter, mine, represent, refine, interact. The COD method fits mainly in the acquire and mine stages. However, there is important work to do in the represent and interact stages to further our understanding of machine learning algorithms. With regard to appropriate representations, recent studies indicate that visualizations that use position are preferred over other forms for comparative numeric data such as COD produces [11]. In particular, line graphs, bar charts and scatterplots are preferred. In experimenting with different visualizations, it is useful to keep in mind that the two most important aspects of visualizations are what question is being answered, and how a user/reader is expected to interact with the data. Otherwise, it is easy to get caught up in the views for their own sakes.

Drummond[12] describes cost curves as a technique to visualize the performance of classifiers based on the distri-

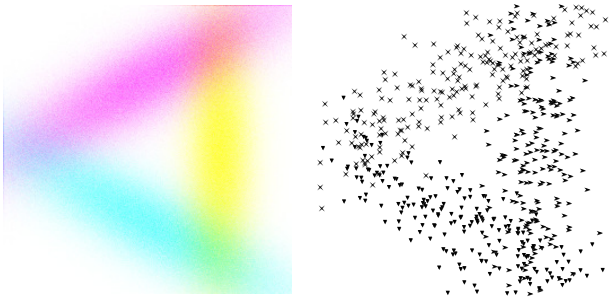

Fig. 1. (Left:) An artificial example of a classification learning task with two real-valued input features and three output classes. (Right:) A grayscale sampling of the three classes plotted with three different monikers.

bution of classes and the cost of misclassifying outputs. By "performance" they mean the *expected cost of misclassifying inputs*. COD, on the other hand, measures *expected dissimilarity*, or the probability that two classifiers will give different outputs for the same input pattern. A misclassification by one algorithm is only significant if it means the output is different from other algorithms. Cost curves and COD measure and visualize different aspects of algorithm behavior.

IV. The COD Distance Between Classifiers

The objective of COD is to measure an estimate of the frequency that two hypotheses, and by extension two learning algorithms, will behave differently on a task or set of tasks. For the distance between two individual hypotheses, we would ideally like to integrate over all possible inputs the difference between the output of the two hypotheses, weighted by the input probability. The ideal COD distance metric over a learning task would be computed something like this:

$$D(h_1, h_2) = \int_{\boldsymbol{x} \in X} |h_1(\boldsymbol{x}) - h_2(\boldsymbol{x})| \cdot \mathrm{P}(\boldsymbol{x}) d\boldsymbol{x} \qquad (1)$$

where $\boldsymbol{x}$ is each possible input from the input space X, the quantity $|h_1(\boldsymbol{x}) - h_2(\boldsymbol{x})|$ is 1 when hypothesis 1 outputs a different value from hypothesis 2 and 0 otherwise, and $\mathrm{P}(\boldsymbol{x})$ is the probability of input $\boldsymbol{x}$ occurring for the task.

Consider the artificial classification problem given in figure 1. It has two real-valued input features, and three output classes. The output classes are normally distributed in space. The left figure is in color, while the right figure is an equivalent grayscale version.

The decision regions for two decision tree hypotheses trained on this problem are given in figure 2.

The region of the input space over which the hypotheses disagree is shown in figure 3. The proportion of the input space covered by this region then forms the basis for the COD metric, after weighting the covered region by the probability of each input occurring, as shown in figure 4.

The integration given above is impractical, or in most cases impossible, for at least two reasons:

1) In most cases, the input space X is of large dimension and/or contains a large (or infinite) number of possible inputs for each feature. Tasks where this is not the case tend to be academic in nature. Although they are properly part of P_I, they form a relatively small portion even when taken together.
2) For many tasks, the likelihood of an input vector for the task ($P(\boldsymbol{x})$) is not known. When it can be estimated, such estimations rely on imperfect or invalid assumptions. Estimation errors can easily compound over a large integration to result in an inaccurate distance metric.

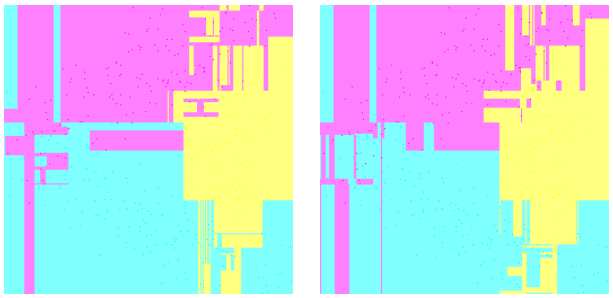

Fig. 2. The decision surfaces for two different hypotheses (generated by two different decision tree learning algorithms) on the problem given in figure 1.

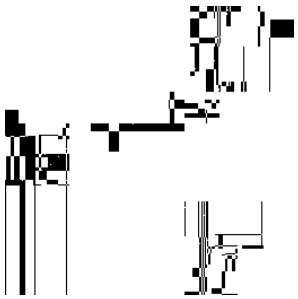

Fig. 3. The region where the two hypotheses from figure 2 disagree.

Instead, the COD measure is calculated using differences in the hypotheses' output behaviors. Two hypotheses are compared using COD by evaluating each hypothesis on patterns that neither has seen during training, and comparing

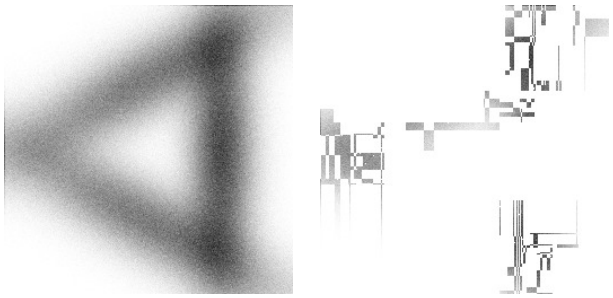

Fig. 4. (Left:) The probability of each input occurring in the input space (darker values are more probable). (Right:) The region from figure 3 weighted by the probability of the input occurring. The proportion of the overall coverage of the right image against the coverage on the left is the COD distance metric.

their outputs. The number of disagreements over the number of patterns used in the comparison yields the COD distance. Taking typical training set/testing set methodology, any test set that is disjoint from the training set may be used to compute the COD distance.

Assuming the test set is representative of the learning task, the COD metric can be estimated as:

$$\hat{D}_T(H_1, H_2) = \frac{\sum_{\boldsymbol{x} \in T} |H_1(\boldsymbol{x}) - H_2(\boldsymbol{x})|}{|T|} \qquad (2)$$

where T is the test set and $|H_1(\boldsymbol{x}) - H_2(\boldsymbol{x})|$ is 1 where hypothesis 1 disagrees with hypothesis 2 on the classification of pattern $\boldsymbol{x}$, and is 0 otherwise.

COD values will reflect differences in output behavior due to differences in how two hypotheses generalize. The distance is nonzero only when the hypotheses have different *observable* behaviors at the output. If two hypotheses give the same answer most of the time, the internal mechanics of how the answers are computed are irrelevant when focusing on output behavior. As "black boxes," such algorithms are equivalent. In such cases, other factors, such as speed or simplicity, may influence the choice of algorithms. In this paper, however, we are only concerned with output behavior.

Instead of using the basic training set/testing set method as a base, n-fold cross-validation may also be used. The COD metric can thus be measured between two algorithms on a task by obtaining a pair of hypotheses, one from each algorithm, over each partitioning of the data set and averaging the pairwise distances between the corresponding hypotheses. This is the method we used to generate our results.

V. Method

The COD measurement has a simple method for combining results on multiple learning tasks to form a single composed measurement. We obtain an estimate of the COD distance for tasks similar to those measured by averaging the COD distances for algorithm pairs on several tasks. This means the results are dependent on the learning tasks chosen to measure the learning algorithms. This is undesirable, but not without precedent in machine learning. For example, measurement on a single learning task is dependent on the particular points sampled from the task. Ideally, to obtain accurate measurements on a task, one performs a large unbiased sampling of the learning task. At the meta-level, selecting the learning tasks on which to measure the COD distance fills the same role here. It is more difficult to obtain a large sampling and bias is more difficult to eliminate. However, even with a somewhat biased sample the most significant trends may still be exhibited.

Table I lists the thirty learning tasks on which the measurements were performed. These tasks are an initial attempt to sample P_I, the set of interesting problems, using tasks from the UCIMLR[3]. Ultimately, it will be desirable to obtain a large number of tasks beyond this repository. The objective was to select a variety of problems of different types, while avoiding artificial data sets.

TABLE I
Learning tasks from the UC Irvine Machine Learning Repository used in these experiments.

Name	Classes	Real inputs	Enum. inputs	Enum. values	Patterns
ann	4	6	15	45	7200
breastw	3	0	9	99	683
bupa	3	6	0	0	345
cmuson	3	60	0	0	208
cmuvow	12	10	0	0	990
dermat	7	1	33	163	358
ecoli	9	7	0	0	336
glass	8	9	0	0	214
iono	3	34	0	0	351
iris	4	4	0	0	150
led7	11	0	7	21	200
lymph	5	0	18	78	148
musk1	3	166	0	0	476
newthy	4	5	0	0	215
pima	3	8	0	0	768
promot	3	0	57	285	106
segm	8	19	0	0	2310
sonar	3	60	0	0	208
splice	4	0	60	540	3190
staust	3	6	8	45	690
stgern	3	24	0	0	1000
stgers	3	7	13	69	1000
sthear	3	5	8	31	270
stsati	7	36	0	0	6435
stsegm	8	19	0	0	2310
stvehi	5	18	0	0	846
vowel	12	10	0	0	528
wave21	4	21	0	0	300
wine	4	13	0	0	178
zoo	8	0	16	52	90

Table II lists the algorithms used in this experiment, along with abbreviations used in the figures. The various algorithms represent eight different families or paradigms. The eight families represented are: Decision Tree (DT), Bayesian (Bsn), Artificial Neural Network (ANN), Instance Based (IB), Support Vector Machine (SVM), Regression (Reg), and Rule Based (Rule). The implementations for nine of the algorithms were taken from the Weka project [13]. The other eight were independently developed.

TABLE II
Learning algorithms and corresponding abbreviations.

Algorithm	Family	Marker
Decision Tree, info. gain [14]	DT	DTg
Decision Tree, gain ratio	DT	DTr
Decision Tree, ratio variant	DT	DTe
Naïve Bayes, Gaussian [15]	Bsn	NB
Naïve Bayes, class segmentation	Bsn	NBs
Naïve Bayes, input segmentation	Bsn	NBi
Single Layer Perceptron [16]	ANN	SLP
Multilayer Perceptron [17]	ANN	MLP
Weka: J48	DT	wJ48
Weka: IB1	IB	w1NN
Weka: IBk (with $k = 5$)	IB	w5NN
Weka: RBFNetwork	ANN	wRBF
Weka: SMO -R	SVM	wSMOr
Weka: SMO	SVM	wSMO
Weka: Logistic	Reg	wLgstc
Weka: JRip	Rule	wJRip
Weka: KStar	IB	wKStar

On each algorithm pair and for each learning task, the COD distance was measured ten times on ten folds which were then averaged together. We considered using a smaller

experiment to illustrate the ideas presented in this paper. However, the concepts and results are not as compelling with fewer algorithms or fewer tasks.

VI. Results and Analysis

The COD measurements between the seventeen algorithms on thirty learning tasks are given in table III. The COD distance is symmetric and the distance between an algorithm and itself is zero. The COD distance also obeys the triangle inequality. Imagine a triangle with any three of the algorithms as the vertices, and the lengths of the sides equal to the corresponding COD values. Any such combination obeys the triangle inequality.

This distance matrix can be used to gain insight into the behavioral relationships between learning algorithms across the algorithm space. Quantifiable measurements of similarity of learning algorithms would be useful in meta-learning, because meta-learners are amenable to numeric representations of the learning algorithms they manipulate. Humans, however, are better suited to representations and visualizations that help filter the data for meaning. Figure 5 shows a heatmap of Table III with distances grouped into four ranges 0-15, 16-21, 22-29, 30-34, which gives a visual representation of the numbers. The most obvious features are that the highest distances are associated with *NBs* and *wSMOr*, and most pairs have a distance between 16 and 21.

The groupings in the heatmap are based on a frequency count of the distances in the table, shown in the bar graph in Figure 6. The smallest distance is 3, between *DTg* and *DTe*, and all other pairs less than 15 are at least 12. The largest distance is 34, the most common distances are 19 and 20.

While these views provide basic statistical information, it is easier to study the relationships among algorithms across the entire space of algorithms using a view that is based on positional information. Suppose, for example, we want to try a number of algorithms on some task and choose the best one. We have tried some number of tasks, and we want to decide which algorithm to try next. Algorithm A performs well on the task, and algorithm B performs poorly. Suppose that algorithm C is similar to A and that algorithm D is similar to B. Similarity suggests that algorithm C should be tried next and that algorithm D should be discounted. Figure 7 shows a two-dimensional scatterplot of the algorithm space for the seventeen learning algorithms tested. This figure was produced by applying a classical multidimensional scaling algorithm to the distance matrix, and plotting the results with gnuplot. Suppose *MLP* performs well and *DTe* does not. Similarity, as defined by distances between points in Figure 7, suggests that we might try *wLgstc* next, but discount trying *DTg* next.

Some observations about the algorithm space, based on this figure, follow. Generally speaking, the algorithms are grouped by family. The decision trees–*DTg*, *DTe*, *DTr*, and *wJ48*–form a group near the upper right corner, for example.

Algorithm families are arranged functionally, although this may be a side-effect of the family grouping. Instance Based algorithms, which tend to focus on local classification, are

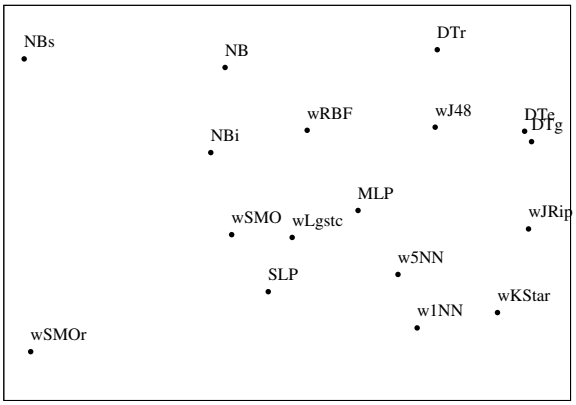

Fig. 7. A two-dimensional rendition of the COD distance matrix.

near the bottom right. Near the center of the cluster is the *MLP*, which generally functions globally but can focus locally when needed. *SLP* and Logistic Regression *wLgstc* are near the *MLP*. *SLP* is similar in operation to the *MLP*, although not as flexible and powerful. Regression also has functional similarities to the Neural Network approaches. These approaches tend to focus on fitting a surface to the task of classification.

The Bayesian algorithms, which function by using statistics gathered globally, are at the top left. And again the decision tree algorithms, which function globally in some regions and locally in others, are in the top right.

The same MDS algorithm could be used to construct three-dimensional representations of these relationships. Three-dimensional visualizations give one extra degree of freedom when dealing with multidimensional data. Thus, they are less constrained than plotting the same data in two dimensions. In general, three dimensional rotations may reveal planes or structures that are obscured in two dimensions. It is also natural to think of the COD matrix as an adjacency matrix for a weighted graph, with each algorithm at a node. In fact, the scatterplot can be viewed as a graph with the edges removed. The results of experiments with graph layout algorithms verified that the scatterplot of Figure 7 is correct.

VII. Other Factors that May Affect Differences

This section adds new information to some additional COD concepts that may affect differences as visualized above. The basic assumption behind the COD metric is that pairs of algorithms with a smaller COD value are similar, and pairs with a larger COD are more different. In the earlier paper [2], Peterson observed that some algorithms are similar even if the COD value is large. Thus the COD values alone may not be sufficient to determine how similar two hypotheses or algorithms are.

One such condition is referred to as *dominance*. The concept of *COD Angle* was developed to measure, detect, visualize and explain dominance [2]. Given two hypotheses h_1 and h_2, h_1 dominates h_2 to the degree that where they disagree, $h1$ is correct and h_2 is incorrect. The COD Angle

TABLE III

DISTANCE MATRIX OF COD DISTANCES (IN PERCENT) BETWEEN ALGORITHMS, AVERAGED OVER THIRTY LEARNING TASKS.

COD	DTe	DTg	DTr	NB	NBs	NBi	MLP	SLP	wLgstc	wRBF	wSMO	wSMOr	w1NN	w5NN	wKStar	wJRip	wJ48
DTe	0	3	17	24	32	20	17	21	19	19	20	33	20	19	20	19	14
DTg	3	0	18	24	32	20	17	21	19	19	20	33	20	19	20	19	14
DTr	17	18	0	25	31	21	18	21	20	20	20	34	21	20	22	21	18
NB	24	24	25	0	26	16	19	21	20	16	19	31	23	21	24	23	22
NBs	32	32	31	26	0	25	28	30	29	27	29	34	32	30	32	31	30
NBi	20	20	21	16	25	0	16	18	18	16	16	28	21	18	21	19	19
MLP	17	17	18	19	28	16	0	13	12	14	12	28	15	13	17	16	16
SLP	21	21	21	21	30	18	13	0	12	17	13	29	19	17	21	19	19
wLgstc	19	19	20	20	29	18	12	12	0	16	12	30	18	17	20	19	18
wRBF	19	19	20	16	27	16	14	17	16	0	16	30	19	16	20	19	18
wSMO	20	20	20	19	29	16	12	13	12	16	0	24	19	16	20	18	18
wSMOr	33	33	34	31	34	28	28	29	30	30	24	0	32	29	33	30	31
w1NN	20	20	21	23	32	21	15	19	18	19	19	32	0	12	13	20	20
w5NN	19	19	20	21	30	18	13	17	17	16	16	29	12	0	15	19	17
wKStar	20	20	22	24	32	21	17	21	20	20	20	33	13	15	0	21	20
wJRip	19	19	21	23	31	19	16	19	19	19	18	30	20	19	21	0	18
wJ48	14	14	18	22	30	19	16	19	18	18	18	31	20	17	20	18	0
Avg.	20	20	22	22	30	20	17	19	19	19	18	31	20	19	21	21	20

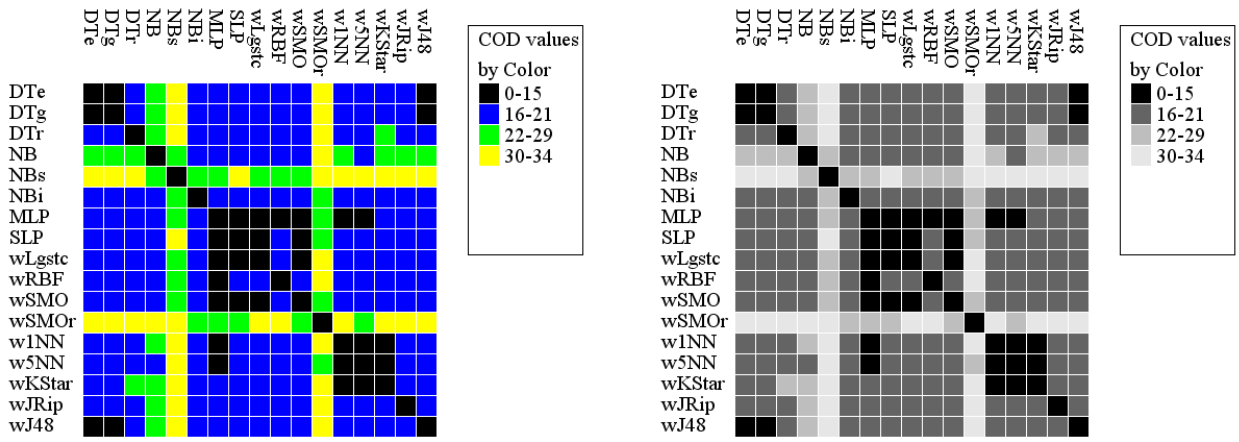

Fig. 5. (Left:) A color heatmap of Table III. (Right:) A grayscale version.

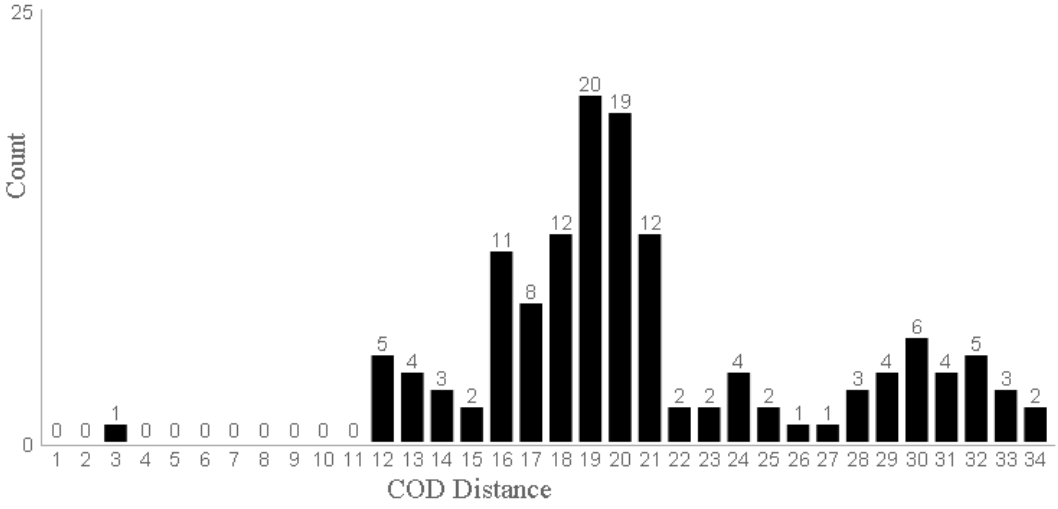

Fig. 6. A frequency count of the distances in Table III.

relates the COD value to the underlying error rates for h_1 and h_2, as a single value. This allows us to view the error rate for a hypotheses or algorithm as the COD between that algorithm and the ideal "perfect classifier", h_0. Recall that h_0 always classifies input patterns correctly.

These ideas, along with the fact that COD distances obey the triangle inequality, allow us to put bounds on those distances. Suppose h_1 and h_2, have respective error rates ϵ_1 and ϵ_2, where the error rate is one minus the accuracy. The following are true because error rate measures the difference between h_0 and any other hypothesis:

$$\hat{D}_T(h_0, h_1) = \epsilon_1 \quad (3)$$

$$\hat{D}_T(h_0, h_2) = \epsilon_2 \quad (4)$$

$$\hat{D}_T(h_1, h_2) \leq \epsilon_1 + \epsilon_2 \quad (5)$$

$$\hat{D}_T(h_1, h_2) \geq |\epsilon_1 - \epsilon_2| \quad (6)$$

Inequality 5 follows from two constraints. First, any input for which h_1 and h_2 give the (same) correct output cannot contribute to ϵ_1, ϵ_2 or D. Second, any input for which h_1 and h_2 give the same incorrect output contributes to ϵ_1 and ϵ_2, but not D. Reaching that upper limit requires h_1 and h_2 to be different for each input where either gives an incorrect output.

Inequality 6 follows by analyzing ϵ_1 and ϵ_2. When ϵ_1 and ϵ_2 are equal, their difference is 0. Since 0 is the smallest possible value for D, the difference is a lower bound. When $\epsilon_1 \neq \epsilon_2$, either $\epsilon_1 > \epsilon_2$, or $\epsilon_2 > \epsilon_1$. Suppose $\epsilon_1 > \epsilon_2$. This implies that there are some instances for which h_2 gives correct output while h_1 gives incorrect output. D must include at least this difference, but may also include other differences. The same argument can be used for $\epsilon_2 > \epsilon_1$. So, $\epsilon_1 - \epsilon_2$ is a lower bound.

It may be useful to formalize the sources of behavioral differences with respect to a training set T. The patterns in a training set can be partitioned into one of five possible sets, when we combine accuracy and output differences. Each description is followed by a two-letter designation, which we can refer to when discussing each set. The partitions are:

1) h_1 and h_2 both give the correct output (CC).
2) h_1 is correct and h_2 is incorrect (CI).
3) h_1 is incorrect and h_2 is correct (IC).
4) h_1 and h_2 both give the same incorrect output (IS).
5) h_1 and h_2 give different incorrect outputs (IX).

Patterns in CC and IS are instances where the two hypotheses agree on the output. These contribute zero to the COD. Thus, the COD measures the proportion of patterns in IC, CI and IX. When counting the differences between hypotheses in order to calculate the COD, it may be useful to keep track of the percentage in each partition as well. These five additional numbers may add richness to a visualization. That is a topic for future work.

A second condition under which two hypotheses with a large COD distance may be similar is when knowing the behavior of h_1 allows us to predict the behavior of h_2. A simple example of this condition occurs for a task with two output classes, in which h_1 always disagrees with h_2. The COD distance is 1, but h_1 is h_2 inverted. In some sense, h_1 and h_2 are learning similar structure, but producing different output values. Space does not permit us to go into detail here, but we have used conditional independence to calculate a value called *expected COD* (ECOD). ECOD measures how likely it is that h_1 and h_2 are independent, based on conditional prior probabilities. In the most general sense, this is an issue of *mimicry*. h_1 mimics h_2 if there exists some function g such that $g(h_1) = h_2$, with acceptable error. Curve-fitting visualizations may be a great aid in refining these particular concepts further.

VIII. Conclusion

The Classifier Output Difference distance metric quantifies the differences in output behavior between pairs of machine learning algorithms. We computed a COD distance matrix for seventeen algorithms, representing eight families, on thirty tasks. We then used the COD data to visualize and analyze the relationships among the eight algorithm families in the algorithm space. Although COD can be used to compare individual algorithms, the focus in this paper is on showing the structure of the algorithm space. A two-dimensional scatterplot was used for this purpose. Results showed the Multilayer Perceptron, which sometimes functions globally, and sometimes locally, near the center. Other algorithm families are grouped around MLP, some more loosely than others. This raises the question of why MLP is in the center.

The uniform numerical representation that the COD method produces lends itself to visualization techniques and automated machine meta-learning algorithms. As we continue to develop interactive visualizations and to explore the algorithm space and the theoretical underpinnings of machine learning, we expect the COD method to be a useful aid to increasing our understanding.

References

[1] T. M. Mitchell, "The need for biases in learning generalizations," Rutgers Computer Science Department, New Brunswick, New Jersey, Tech. Rep. CBM-TR-117, May 1980.

[2] A. H. Peterson and T. R. Martinez, "Estimating the potential for combining learning models," in *Proceedings of the ICML Workshop on Meta-Learning*, 2005, pp. 68–75.

[3] A. Asuncion and D. Newman, "UCI machine learning repository," 2007. [Online]. Available: http://www.ics.uci.edu/∼mlearn/MLRepository.html

[4] J. B. Kruskal and M. Wish, *Multidimensional Scaling*, ser. Quantitative Applications in the Social Sciences. Sage, 1978, no. 11.

[5] D. H. Wolpert, "The supervised learning no-free-lunch theorems," in *Proceedings of the 6th On-line World Conference on Soft Computing in Industrial Applications*, ser. Springer Engineering Series, 2001.

[6] L. I. Kuncheva and C. J. Whitaker, "Measures of diversity in classifier ensembles," *Machine Learning*, no. 51, pp. 181–207, 2003.

[7] B. Pfahringer, H. Bensusan, and C. Giraud-Carrier, "Meta-learning by landmarking various learning algorithms," in *Proceedings of the Seventeenth International Conference on Machine Learning, ICML'2000*. Morgan Kaufmann, San Francisco, California, 2000, pp. 743–750.

[8] Z. Zheng, "A benchmark for classifier learning," Basser Department of Computer Science, N.S.W Australia 2006, Tech. Rep. TR474, 1993.

[9] C. E. Brodley, "Dynamic automatic model selection," University of Massachusetts, Tech. Rep. UM-CS-1992-030, February 1992.

[10] B. Fry, *Visualizing Data*, 1st ed., 2008.

[11] J. Heer, M. Bostock, and V. Ogievetsky, "A tour through the visualization zoo," *Commun. ACM*, vol. 53, pp. 59–67, June 2010. [Online]. Available: http://doi.acm.org/10.1145/1743546.1743567
[12] C. Drummond and R. C. Holte, "Cost curves: an improved method for visualizing classifier performance," in *Machine Learning*, 2006, pp. 95–130.
[13] I. H. Witten and E. Frank, *Data Mining: Practical machine learning tools and techniques*, 2nd ed. San Francisco: Morgan Kaufmann, 2005.
[14] J. R. Quinlan, *C4.5: Programs For Machine Learning*. Morgan Kaufmann Publishers, Inc., 1993.
[15] K. Lang, "NewsWeeder: learning to filter netnews," in *Proceedings of the 12th International Conference on Machine Learning*. Morgan Kaufmann publishers Inc.: San Mateo, California, 1995, pp. 331–339.
[16] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychological Review*, vol. 65, pp. 386–408, 1958.
[17] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," *Parallel distributed processing: explorations in the microstructure of cognition, vol. 1: foundations*, pp. 318–362, 1986.