Intelligent Data Analysis 14 (2010) 355–367
DOI 10.3233/IDA-2010-0425
IOS Press

Using learning algorithm behavior to chart task space: The DICES distance

Adam H. Peterson[*] and Tony R. Martinez
Computer Science Department, Brigham Young University, Provo, UT, USA

Abstract. In theory, learning is not possible over all tasks in general. In practice, the tasks for which learning is desired exhibit significant regularity, which makes learning practical. For the most effective learning, it is valuable to understand the nature of this regularity and how it manifests in the tasks where learning is applied. This research presents the DICES distance metric for finding similarity between learning tasks. With this distance metric, a collection of learning tasks can be given a distance matrix. This distance matrix can be used for visualizing the relationships between learning tasks and searching through task space for tasks which are similar in nature. Examples of task visualization are given, and other possible applications of this tool are touched upon. Such applications include learning algorithm selection, transfer learning, and analysis of empirical results.

Keywords: Machine learning, meta learning, task similarity, task space

1. Introduction

Wolpert's *no free lunch theorem* indicates that in a general sense learning is not possible [11] Learning algorithms which generalize in a way that improves performance for one class of functions will, at the same time, impair performance for another class. Nevertheless, machine learning has found consistent and significant use in research and industry. The premise of the no free lunch theorem which treats functions as equally likely does not hold in practical application. The class of tasks to which machine learning is routinely applied is much narrower than the set of all functions. Learnable functions correspond at least indirectly to the operations of mechanisms in the real world, and exhibit a large degree of regularity and structure. It is therefore valuable for machine learning approaches to be designed with an understanding of the nature of this collection of learnable functions.

This research presents a method to measure differences in learning tasks as a tool for understanding the relationships between tasks. This tool is used to find a distance matrix between learning tasks drawn from the Irvine Machine Learning Repository. Visualization of the spatial relationships between these tasks is also shown, and the tasks are grouped in clusters. Observations about these tasks are then made using these tools for examining their relationships with each other.

Although the primary purpose of this research is to examine and understand the nature of learnable tasks and their relationships, this distance metric can be used for other practical purposes. When machine learning research is presented, especially on a new learning algorithm or a variation of an existing one, it is common to validate the research with tables of empirical results. Often the research makes either an overt or implied claim that the proposed algorithm performs well over a diverse collection of learning

[*]Corresponding author. E-mail: adam@axon.cs.byu.edu.

tasks, and validates it by giving empirical results showing the algorithm performing favorably on the bulk of learning tasks. If care is not taken when selecting learning tasks to ensure that one type of learning task is not over-represented, the results will not actually substantiate this implied claim. Even if the researcher puts forth an effort to use a diverse collection of learning tasks, it is not always easy to determine whether two learning tasks are similar in nature and thus potentially redundant. With a better understanding of how learning tasks are distributed and related, researchers can select tasks with the objective of thoroughly covering the field of learnable tasks.

At other times, a researcher's goal may be the opposite. Having constructed a new algorithm or method, she carefully determines or hypothesizes what types of learning tasks will benefit from it, and wishes to gather a collection of learning tasks of this nature. In this case, the same difficulty occurs in reverse—it is not always evident whether two learning tasks are similar. It is desirable to be able to find learning tasks which are related, and the research presented here can provide tools for doing this.

2. Related work

Prior work in the area of relating learning tasks includes work by Pfahringer on *Landmarking*. Landmarking positions tasks in a task space by executing simple learning models on them and measuring the performance of the models. The research presented herein is inspired by Landmarking, although tasks are related by the *behavior* of learning models, rather than the *performance* of models as in Landmarking. In the Landmarking system, rules are induced (e.g. if algorithm A does better than algorithm B, then algorithm C will do better than algorithm D) and the objective is to determine which learning algorithms to employ based on the performance of these markers. The focus of the research presented here is on finding relationships and understanding the structure of "task-space" and how tasks relate to each other.

The task of matching learning algorithms to problems is approached by Brodley [3]. Brodley's work uses ambitious heuristics to try to match a learning model to a task. As with Landmarking, Brodley's work is focused on producing rules and heuristics to match a learning task to an appropriate algorithm to maximize accuracy. Brodley measures characteristics of the learning task rather than using simple algorithms as markers.

One of the ultimate goals of this research is to provide a more robust and sound way of supporting research through empirical results. Aha [1] scrutinizes empirical case studies with a similar goal. Aha's emphasis is to encourage the use of a more rigorous framework for case studies to improve the quality of the results being presented.

The concept of task similarity is valuable in the context of transfer learning and lifelong learning, such as is given by [9,10] Thrun demonstrates that learning incrementally across multiple task domains is effective. Thrun also presents a hierarchical clustering approach for learning tasks called *TC*, or Task Clustering [8] Thrun's TC approach is focused on transfer learning and clusters learning tasks based on the potential for a model trained on one task to be able to effectively represent another.

Baxter [2] discusses lifelong learning with an emphasis on using related learning tasks. Baxter posits that lifelong learning is most effective when the same internal representation can be used for several related tasks. The distance metric presented in this research can be used as a tool to discover and quantify the relatedness of tasks based on their learnable regularity.

3. Measuring learning task similarity

The tasks toward which machine learning is applied all have some degree of regularity and structure to them. It is this structure which makes learning possible. Learning algorithms exploit the structure to perform predictions and classifications. Many learning algorithms have been developed in machine learning over several decades. Some algorithms recognize one type of structure, while others recognize another. It is an open question whether all types of regularity occurring in learnable tasks have been exploited by some algorithm, and it will probably remain an open question for the foreseeable future. Regardless, a sizable collection of diverse learning algorithms with different strengths and weaknesses is available today.

Different professionals approach machine learning with different goals. Some have a specific set of learning tasks that are important to them and are always looking for ways to improve performance on these particular tasks. Others are researchers, working to create and improve learning algorithms to cover a wide variety of learning tasks and achieve improved performance over them generally. Yet other researchers are interested in creating and reasoning about cognitive models that exhibit general problem solving and adaptability inspired by human intelligence. As different people have different goals, it can be difficult for one practitioner to know when the published work of another is relevant to her own objectives. A method to determine how related two learning tasks are can be a valuable tool for a professional to evaluate the usefulness of published results.

More generally, it is valuable to understand the relationship between learning tasks in order to better understand what makes them learnable. With better understanding of how learnable tasks relate to each other and how they differ from the general class of functions Wolpert discusses as unlearnable, the task of learning in general can be approached more effectively.

This research provides such a tool for determining the relationships between learning tasks.

3.1. The COD metric for algorithm behavior variation

The distance metric for learning tasks presented in this research uses the COD (Classifier Output Difference) distance measure for learning algorithms by Peterson [6] as a tool. The problem of learning task similarity is approached here by examining the effect learning tasks have on the behavior of the models produced by learning algorithms. Given a learning task and two learners, the COD metric is a measure of the extent to which the learners produce differing predictions. More specifically, it is an estimate of the probability that, given a new instance, the learners will classify the instance differently. Algorithm 1 describes how the COD distance is estimated. In essence, the two algorithms are used to produce hypotheses from a classification training set, and the frequency of disagreement of those algorithms on the test set is measured.

3.2. From algorithm diversity to learning task diversity

The core contribution of the research given here is a way to characterize and measure the similarity and difference between learning tasks from a learnability standpoint. Tasks are considered similar if they elicit similar relative behavior characteristics from learning algorithms. The COD distance is used to quantify the behavior characteristics of a task with respect to a learning algorithm. When two tasks have similar observed effects on the models produced by various algorithms, this provides evidence that the tasks have similar types of learnability.

For example, if task t_1 results in a small COD distance between algorithms a_1 and a_2, but a large distance between algorithms a_2 and a_3, while task t_2 similarly gives a small COD distance between a_1 and a_2 and a large distance between a_2 and a_3, this indicates that the tasks bear similarity to each other in terms of their "learnable potential." (Recall that when using the COD metric, algorithm distances are measures of behavior difference.) If, on the other hand, t_3 gives a large distance between a_1 and a_2, but

Algorithm 1 Estimating the COD metric

```
function ESTIMATE_COD($T$, $a_1$, $a_2$)    ▷ Estimate the COD metric between
                                              ▷ two algorithms using a data set
    $F = \text{partition}(T)$                   ▷ Partition the data into folds
    for $a \in \{a_1, a_2\}$ do                  ▷ For each of the two algorithms
        for all $f \in F$ do                    ▷ For each fold of the data set
            $H_{a,f} = \text{train}(a, T - f)$      ▷ Train a hypothesis using all the data
                                              ▷ except the current fold
            $L_{a,f} = \{p \rightarrow H_{a,f}(p)\}$   ▷ Use this hypothesis to label the patterns
                                              ▷ in this fold
        $L_a = \{L_{a,f}\}$   ▷ Combine the labels into a single set for this algorithm
    return $\frac{\left|\left\{p \in T | L_{a_1,p} \neq L_{a_2,p}\right\}\right|}{|T|}$   ▷ Return the fraction of patterns $a_1$ and $a_2$
                                              ▷ disagree on
```

Algorithm 2 Estimating the DICES distance between tasks

```
function ESTIMATE_DICES($T_1$, $T_2$, $A = \{a_i\}$)      ▷ Estimate the DICES
        ▷ distance between two data sets (representing learning tasks) with
                                 ▷ respect to a set of learning algorithms
    for $T \in \{T_1, T_2\}$ do                       ▷ For each of the two tasks
        for all $\{a_i, a_j\} \in A \times A$ do          ▷ And over each pair of algorithms
            $C_{ij}(T) = \text{Estimate\_COD}(T, a_i, a_j)$      ▷ Find the COD distance
        $\hat{\mathbf{c}}_T = \left[C_{12}(T), C_{13}(T), \cdots, C_{1n}(T), \cdots, C_{23}(T), \cdots, C_{(n-1)n}(T)\right]$
                        ▷ Arrange the COD distances into a vector for task $T$
        $\mathbf{c}_T = \frac{\hat{\mathbf{c}}_T}{|\hat{\mathbf{c}}_T|}$        ▷ Normalize the COD distance vector to unit length
    return $|\mathbf{c}_{T_1} - \mathbf{c}_{T_2}|$
```

a small distance between a_2 and a_3, this indicates that there is something about learning task t_3 that is different in character from t_1 and t_2. In this research, relative COD distances are compared rather than absolute distances. This is done because the COD distance will have a tendency to be smaller for tasks that are simpler (have less process noise, fewer outliers, etc.) as a consequence of there being fewer missed patterns for algorithms to disagree on. Using relative distances can reduce this effect.

One may observe that the term "learnable potential" used here does not have a strict definition. Conceptually, this term relates to the aspects of a task that make it learnable, the qualities of regularity contained within it. This term can not be defined formally; otherwise, the definition could be used to characterize the circumstances under which learning is possible, which does not unify well with Wolpert's No Free Lunch theorem. However, although the learnable character of a learning task may be difficult to formulate, it is still valuable to make an effort to understand some of its properties and aspects. By observing which tasks produce similar effects on COD algorithm distances and which produce differing effects, one can begin to quantify the similarity of learning tasks. This quantification can lead to observing learning tasks in quasi-spatial relationships.

We now present the Difference In COD Estimate Similarities (DICES) distance measurement between learning tasks. The procedure for computing the DICES metric between two tasks is given in Algorithm 2, and outlined here:

1. A set of COD distances between algorithms on each task is computed. The COD distance between

algorithms a_1 and a_2, members of a set of algorithms A, over a task t is denoted as $C_{a_1,a_2}(t)$.

2. These pairwise COD distances are then placed a vector of distances for each task:

$$\hat{\mathbf{c}}_A(t) = [C_{a_1,a_2}(t), C_{a_1,a_3}(t), \cdots, C_{a_1,a_n}(t), C_{a_2,a_3}(t), \cdots, C_{a_2,a_n}(t), \cdots, C_{a_{n-1}(t),a_n}]$$

   where $n = |A|$.

3. The vector of distances is then normalized:

$$\mathbf{c}_A(t) = \frac{\hat{\mathbf{c}}_A(t)}{|\hat{\mathbf{c}}_A(t)|}$$

4. Finally, the distance between the two tasks t_1 and t_2 is calculated by measuring the length of the difference between their normalized COD vectors:

$$D_A(t_1, t_2) = |\mathbf{c}_A(t_1) - \mathbf{c}_A(t_2)| \tag{1}$$

The intuitive basis for measuring learning task similarity in this manner is related to the idea that learnable problems exhibit regularity and structure that is exploited by learning algorithms to improve predictive accuracy. It is this structure in a learning task which makes a task learnable. Without some amount of regularity and structure in a task, the premise of the No Free Lunch Theorem applies. No assumptions can be made about the task, and the task can not be effectively learned. Learnable tasks, on the other hand, have significant structure that can be detected by learning algorithms and used to perform generalization.

Different kinds of structure are found in different learning tasks, and different learning algorithms exploit this regularity in different ways. For example, if one is given the learning task of predicting how crowded a swimming pool will be from the outside temperature that day, a linear learning model such as a perceptron will attempt to determine a correlation between temperature and patronage, and output a model that predicts large crowds on one side of a threshold and small crowds on the other. An instance based model, on the other hand, will use the premise that past examples will likely be similar to future instances with similar temperatures, and will make a prediction based on previous days with similar temperatures. Although both learning algorithm types are able to produce a model for predicting the target, these models function in different ways, exploiting the regularity of the target problem differently.

In some cases, the different ways two algorithms exploit task regularity will lead to models with very different behaviors. In other cases, the difference in model representation will be less relevant and still result in learning models that have similar behaviors. Because different models represent different kinds of regularity and in different ways, these behavioral differences between models can serve as an indirect indicator of what types of regularity a learning task contains. Intuitively, if two learning tasks have regularity of a similar nature to one another, different learning algorithms will capture this regularity in each task in a similar way. We can infer similarity between two tasks from similar patterns in the COD measurements they induce in learning algorithms.

The DICES distance between a task and itself will always be zero, since each learning algorithm (as executed on deterministic hardware) will induce the same hypothesis when given the same data set. Some learning algorithms, such as multilayer perceptrons, Boltzmann machines, and random forests, employ pseudo-random sources to set initial conditions or govern other aspects of the training process. Algorithms with high variance can exhibit a high sensitivity to resampling or presentation order. By varying random sources, sampling, and presentation order, algorithms may induce hypotheses with nonzero COD distances to each other. The characteristics of these distances and the resulting DICES distances are potentially interesting, but beyond the scope of this work.

In this research, a learning task is characterized by a data set drawn from the task. There are domains of machine learning in which the data set is not constant, such as Active Learning [4] where the learning algorithm has the option of querying an expert on unseen patterns. Such domains are interesting, but they are more difficult to characterize succinctly, and are thus beyond the scope of this research. Applying the DICES distance in such domains may be a future research direction, if the COD distance (or an analog) can be formulated between Active Learning tasks or other tasks not directly characterizable by a data sampling.

Many learning tasks have some statistical dependencies in their input features. This can occur through various mechanisms, but a primary effect is some information redundancy across features. Na ïve Bayes is a classical example of a learning algorithm for which conditional dependence tends to impair accuracy. Many other learning algorithms are more robust to this occurrence. Decision Trees, for example, will select the features that expose the most new information, discounting features later in training that have much the same information as features already used. Neural Networks using Backprop, on the other hand, will iteratively update the learning model in the direction of greatest error reduction, which will use redundant features in combination until their contribution to the gradient dissipates. These different approaches to learning, with different behaviors surrounding attribute dependencies, will produce hypotheses with likewise different behaviors. By using several algorithms to measure the DICES distance between tasks, covering several different responses to such attribute dependencies, tasks with similar attribute dependencies will have similar DICES vectors and a lower DICES distance.

4. Results and analysis

With the DICES measure, a distance matrix can be constructed for a collection of tasks, and visualization and clustering of the task space can be performed. Table 1 lists 30 tasks from the Irvine Machine Learning Repository [5] over which DICES distances were measured. The 18 learning algorithms used for these DICES measures are listed in Table 2.

The computed DICES distances between the tasks are given in Table 3. The average distance between two tasks is 0.580 and the table has bolded entries for any distance less than 0.365, one standard deviation less than the average. One quirk of this collection of data sets, which is an example of why examining redundancy in a collection of data sets can be useful, is that four pairs of data sets are each essentially the same task, with either a change in the way the inputs or outputs are encoded, or the sampling order. The *stgern* and *stgers* tasks are the same except that in the second some of the inputs have been re-encoded as continuous rather than nominal. The *cmuvow* and *vowel* tasks are the same task except that the output has been encoded as a 1-to-10 discrete value in one and a 1-of-10 nominal value in the other, and in one data set about 40% of the patterns have been removed. The tasks *stsegm* and *segm* are the same task, except that in one data set, the output has been coded as a nominal value with seven classes, and in the other the patterns are marked with a discrete value between 1 to 7. The tasks *cmuson* and *sonar* are both the rocks-and-mines problem, with the only difference being that in one the rock is class 1 and in the other the mine is class 1.

These observations are all reflected in the DICES distance matrix. Four of the five shortest distances are the four task pairs mentioned. The shortest distance is 0.014, between *stsegm* and *segm*. The distance between *cmuson* and *sonar* is 0.066, the second shortest distance in the table. The distance between *cmuvow* and *vowel* is 0.140, which is the third shortest distance of the table. The fifth shortest distance in the matrix is 0.194 between *stgern* and *stgers*. All four of these distances are well below the 0.365 value of the mean minus a standard deviation.

Table 1
Learning tasks used in this research and some of their characteristics

Data Set	Instances	Classes	Features	Bool.	Enum.	Disc.	Cont.
ann	7200	3	21	15			6
breastw	699	2	9			9	
bupa	345	2	6				6
cmuson	208	2	60				60
cmuvow	990	6	10			10	
dermat	366	6	34	1		32	1
ecoli	336	8	7				7
glass	214	7	9				9
iono	351	2	34				34
iris	150	3	4				4
led7	200	10	7	7			
lymph	148	4	18	9		9	
musk1	476	2	166				166
newthy	215	3	5				5
pima	768	2	8				8
promot	106	2	57		57		
segm	2310	7	19				19
sonar	208	2	60				60
splice	3190	3	60		60		
staust	690	2	14	4		4	6
stgern	1000	2	24				24
stgers	1000	2	20		13		7
sthear	270	2	13	3	1	4	5
stsati	6435	6	36				36
stsegm	2310	7	19				19
stvehi	846	4	18				18
vowel	990	6	10			10	
wave21	300	3	21				21
wine	178	3	13				13
zoo	101	7	16	15	1		

Columns in order are: Data set name, instance count, output class count, input feature count, boolean inputs, enumerated inputs, discrete inputs, and continuous inputs.

Three statements can be made from the DICES distance measurements for these task pairs. First, although for each of these pairs the two tasks have essentially the same abstract structure, the hypotheses produced by the various algorithms had some behavior differences. The properties of learning tasks are to some degree sensitive to specific choices made about encoding.

Second, our current learning algorithms are perhaps not as “smart” as might be wished. The behavior of the hypotheses each algorithm produces is not only governed by the principles and theory that went into the design of the algorithm itself, but also by the choices and assumptions of the implementer in deciding how to transform data into a format the learning algorithm can consume. Ideally, a learning algorithm would not suffer for seemingly innocuous choices about encoding such as whether an input is discrete or continuous.

Third, although not negligible, the DICES distances for these task pairs were substantially lower than for most pairs. The DICES metric can be effectively used to find similar learning tasks.

Another observation that can be made is that the data sets tend to form cliques. For example, the *glass* data set is near the *bupa*, *cmuson*, *cmuvow*, *sonar*, and *shvehi* data sets. Moving over to the *bupa* column, it can be seen that this data set is also close to *cmuson*, *glass*, *sonar*, *staust*, and *stvehi*. Three of the four tasks closest to *glass* (not counting *bupa*) are also three of the four closest tasks to *bupa* (not counting

Table 2
List of algorithms used for the DICES measurements

Algorithm	Family
C4.5	Decision tree
ID3 expect	Decision tree
ID3 gain	Decision tree
ID3 ratio	Decision tree
1-NN	Instance based
5-NN	Instance based
KStar	Instance based
Logistic	Regression
JRip	Rule based
NB Gaussian	Naïve Bayes
NB class seg	Naïve Bayes
NB input seg	Naïve Bayes
NB Weka	Naïve Bayes
MLP	Neural network
RBF	Neural network
SLP	Neural network
SVM	Support vector
SVM radial	Support vector

glass).

Another clique can be observed by looking at the near neighbors of *pima*, which are: *cmuson*, *sonar*, *stgern*, *stgers*, and *wave21*. The neighbors of *wave21* are: *cmuson*, *pima*, *sonar*, *stgern*, and *stgers*. In this case, all four of the nearest neighbors are common between these two tasks, *pima*, and *wave21*.

4.1. Visualizing task relatonships

It is insightful to observe a visual representation of relationships between entities. Although the DICES metric is a Euclidean distance measure, this Euclidean space is of very high dimension. Because there are 18 algorithms, there are $\frac{17 \cdot 18}{2} = 154$ components to the vector, which means that the distance is actually in a 154 dimensional space. However, as there are only 30 tasks to visualize, a 29-dimensional subspace will actually suffice to represent all distances with full fidelity. Unfortunately, 29 dimensions is still much higher than we have convenient visualization tools. Still, an approximate visualization can be shown in three dimensional space, by placing points representing tasks into 3-space and then updating their positions with gradient-descent to match the distance matrix.

Such a visualization is shown in Fig. 1. In this figure, the nodes have been given colorations to help distinguish the structure of the representation. The coloration is chosen based on average distance to the rest of the tasks, with darker nodes having shorter average distances to all other tasks and lighter nodes having longer average distances.

The visualization shows, as one would expect, the four "pair" (*stgern* and *stgers*, *cmuson* and *sonar*, *stsegm* and *segm*, and *cmuvow* and *vowel*) tasks close together. In fact, *stsegm* and *segm* are virtually on top of each other, as are *cmuson* and *sonar*. On the other hand, there are several tasks which are strongly distinctive, which tend to hover on the outskirts of the space. The overall layout of this space appears to be a dense cluster of tasks in the middle of the space and a number of other tasks loosely grouped around it.

By viewing this learning task structure in three dimensions instead of 29, we do lose some information. It is useful to consider how much information is being lost. For some insight into this issue, one can look

Table 3
The distance between each learning task, using the DICES metric. The table is symmetric, so only the lower half is shown. It is split horizontally to fit on the page. The mean distance between two tasks is 0.580, and the standard deviation is 0.215

		(**AN**)	(**BR**)	(**BU**)	(**CS**)	(**CV**)	(**DE**)	(**EC**)	(**GL**)	(**IO**)	(**IR**)	(**L7**)	(**LY**)	(**M1**)	(**NE**)	(**PI**)
ann	(**AN**)															
breastw	(**BR**)	0.605														
bupa	(**BU**)	0.466	0.495													
cmuson	(**CS**)	0.497	0.442	**0.215**												
cmuvow	(**CV**)	0.549	0.585	0.392	0.366											
dermat	(**DE**)	0.597	**0.327**	0.419	0.386	0.537										
ecoli	(**EC**)	0.707	0.791	0.578	0.552	0.499	0.679									
glass	(**GL**)	0.467	0.567	**0.248**	**0.272**	**0.283**	0.515	0.504								
iono	(**IO**)	0.831	0.875	0.666	0.657	0.725	0.745	0.558	0.739							
iris	(**IR**)	0.688	0.760	0.581	0.545	0.460	0.642	0.399	0.548	0.506						
led7	(**L7**)	0.692	0.868	0.738	0.729	0.547	0.841	0.627	0.581	0.955	0.666					
lymph	(**LY**)	0.539	**0.288**	0.392	**0.333**	0.498	**0.363**	0.715	0.451	0.823	0.686	0.794				
musk1	(**M1**)	0.551	0.564	0.465	0.453	0.596	0.548	0.701	0.511	0.774	0.722	0.847	0.518			
newthy	(**NE**)	0.658	0.813	0.655	0.648	0.470	0.779	0.541	0.502	0.879	0.588	**0.260**	0.746	0.782		
pima	(**PI**)	0.457	0.485	**0.342**	**0.264**	0.403	0.429	0.508	**0.339**	0.707	0.504	0.632	0.376	0.452	0.571	
promot	(**PR**)	0.524	**0.359**	0.452	0.412	0.582	0.386	0.755	0.521	0.844	0.717	0.838	**0.329**	0.519	0.810	0.395
segm	(**SE**)	0.965	1.005	0.789	0.785	0.835	0.869	0.607	0.847	**0.240**	0.611	1.041	0.953	0.900	0.972	0.838
sonar	(**SO**)	0.498	0.439	**0.218**	**0.066**	0.368	0.390	0.549	**0.277**	0.661	0.557	0.741	**0.329**	0.453	0.661	**0.273**
splice	(**SP**)	0.528	0.567	0.596	0.599	0.762	0.643	0.875	0.663	0.935	0.881	0.893	0.540	0.535	0.899	0.558
staust	(**SA**)	0.528	0.539	**0.282**	**0.309**	0.515	0.477	0.676	**0.358**	0.777	0.693	0.810	0.386	0.427	0.749	0.393
stgern	(**SN**)	0.467	0.424	**0.325**	**0.256**	0.457	0.391	0.596	0.369	0.761	0.600	0.741	**0.325**	0.443	0.686	**0.194**
stgers	(**SS**)	0.482	0.405	**0.362**	**0.302**	0.471	0.386	0.662	0.408	0.797	0.635	0.721	**0.282**	0.423	0.676	**0.234**
sthear	(**SH**)	0.547	0.368	0.436	0.382	0.581	0.364	0.738	0.511	0.844	0.699	0.835	**0.311**	0.445	0.774	**0.334**
stsati	(**ST**)	0.578	0.535	**0.294**	**0.249**	0.411	0.408	0.507	0.390	0.499	0.464	0.791	0.429	0.503	0.718	**0.352**
stsegm	(**SG**)	0.965	1.006	0.790	0.787	0.836	0.871	0.608	0.848	**0.240**	0.611	1.043	0.954	0.901	0.973	0.839
stvehi	(**SV**)	0.516	0.532	**0.282**	**0.262**	**0.320**	0.456	0.497	**0.239**	0.685	0.537	0.661	0.427	0.493	0.609	**0.353**
vowel	(**VO**)	0.574	0.667	0.467	0.446	**0.140**	0.616	0.484	**0.313**	0.764	0.476	0.444	0.581	0.650	0.367	0.450
wave21	(**WV**)	0.462	**0.351**	**0.326**	**0.267**	0.450	**0.319**	0.601	0.402	0.701	0.565	0.744	**0.297**	0.385	0.680	**0.245**
wine	(**WI**)	0.836	1.006	0.860	0.840	0.646	0.954	0.595	0.708	0.929	0.642	**0.297**	0.944	0.942	**0.319**	0.749
zoo	(**ZO**)	0.694	0.759	0.663	0.638	0.448	0.697	0.532	0.522	0.869	0.571	**0.323**	0.704	0.786	**0.320**	0.588

	(**PR**)	(**SE**)	(**SO**)	(**SP**)	(**SA**)	(**SN**)	(**SS**)	(**SH**)	(**ST**)	(**SG**)	(**SV**)	(**VO**)	(**WV**)	(**WI**)	(**ZO**)
segm	(**SE**)	0.970													
sonar	(**SO**)	0.413	0.784												
splice	(**SP**)	0.426	1.063	0.599											
staust	(**SA**)	0.461	0.883	**0.307**	0.585										
stgern	(**SN**)	**0.308**	0.891	**0.256**	0.500	**0.347**									
stgers	(**SS**)	**0.317**	0.932	**0.312**	0.507	**0.334**	**0.199**								
sthear	(**SH**)	**0.330**	0.974	0.388	0.543	0.382	**0.279**	**0.215**							
stsati	(**ST**)	0.483	0.620	**0.245**	0.678	0.397	0.372	0.410	0.474						
stsegm	(**SG**)	0.971	**0.014**	0.785	1.064	0.884	0.891	0.933	0.975	0.621					
stvehi	(**SV**)	0.490	0.787	**0.258**	0.652	0.380	0.374	0.412	0.488	**0.315**	0.789				
vowel	(**VO**)	0.658	0.864	0.453	0.812	0.582	0.530	0.537	0.654	0.501	0.865	0.384			
wave21	(**WV**)	**0.323**	0.845	**0.272**	0.504	0.392	**0.241**	**0.245**	**0.264**	**0.335**	0.846	**0.351**	0.526		
wine	(**WI**)	0.984	0.989	0.852	1.047	0.938	0.872	0.870	0.966	0.873	0.990	0.774	0.535	0.867	
zoo	(**ZO**)	0.765	0.960	0.649	0.891	0.750	0.679	0.665	0.764	0.694	0.962	0.569	0.369	0.666	0.383

at the eigensystem of the covariance matrix of the higher dimensional space. The sorted eigenvalues are shown in Fig. 2. Here, one can see that there are two strong eigenvalues, representing two dimensions of high information content, followed by a number of rather small eigenvalues. From this it can be inferred that a two dimensional representation is enough to capture a large amount of the structure. By using a three dimensional visualization, two dimensions can capture the principal dimensions of the

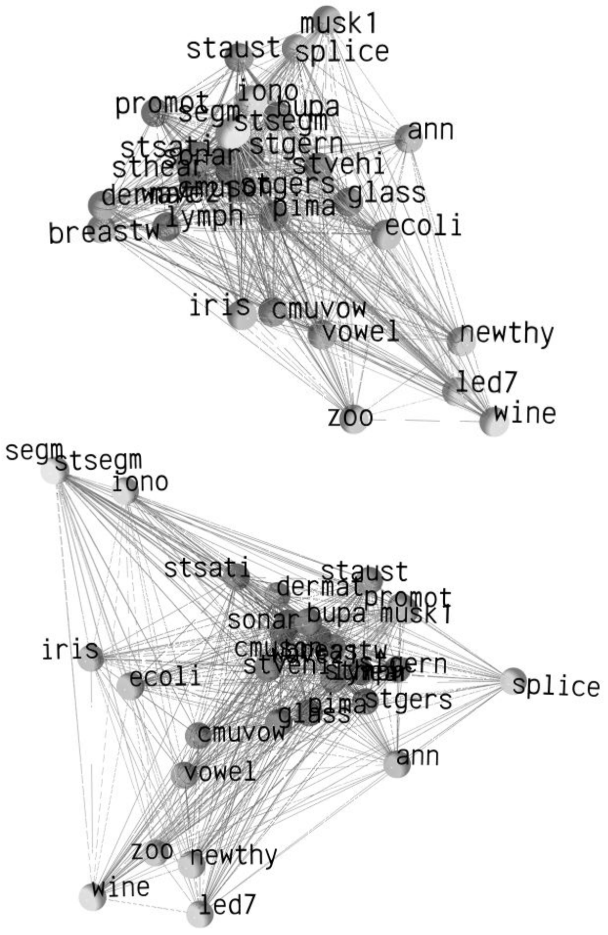

Fig. 1. A visualization of task space in three dimensions for thirty tasks from the Irvine Machine Learning Repository. Two views of the model are shown, roughly 90 degrees apart.

arrangement and the third dimension can allow some flexibility in node placement to retain the local spatial relationships.

4.2. Task relationships through clustering

With a distance matrix over learning tasks, it is possible to do various forms of analysis on the task relationships, such as cluster analysis. Figure 3 shows a dendrogram of learning tasks from doing average link agglomerative clustering. The nodes of this figure are colored to match the colors used in Fig. 1. In this figure, we can observe that the cliques that were noted in the beginning of section 4 have formed clusters. The first clique that was mentioned consisted of *glass*, *bupa*, *cmuson*, *sonar*, and *stvehi*. In Fig. 3, we observe that with the addition of *stsati* these tasks all form a cluster. The second clique observed consisted of *pima*, *cmuson*, *sonar*, *stgern*, *stgers*, and *wave21*. Although *cmuson* and *sonar* are part of the first cluster, the other four members of this clique form a cluster in the dendrogram.

We may also observe in this dendrogram that the darker central nodes agglomerate to form a single large cluster, with the lighter exterior nodes forming their own clusters. It appears that in the collection

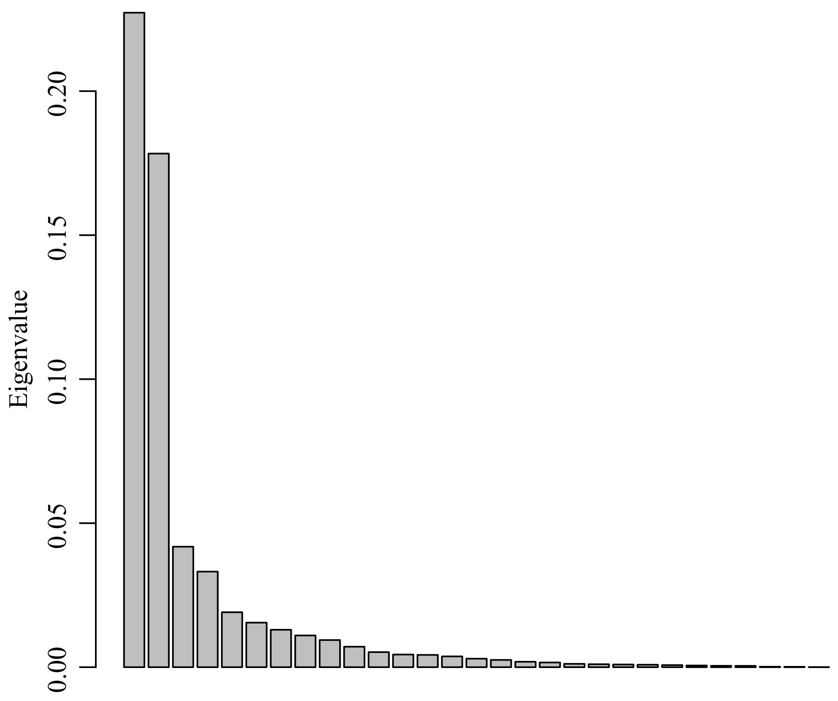

Fig. 2. The eigenvalues of the task space's covariance matrix.

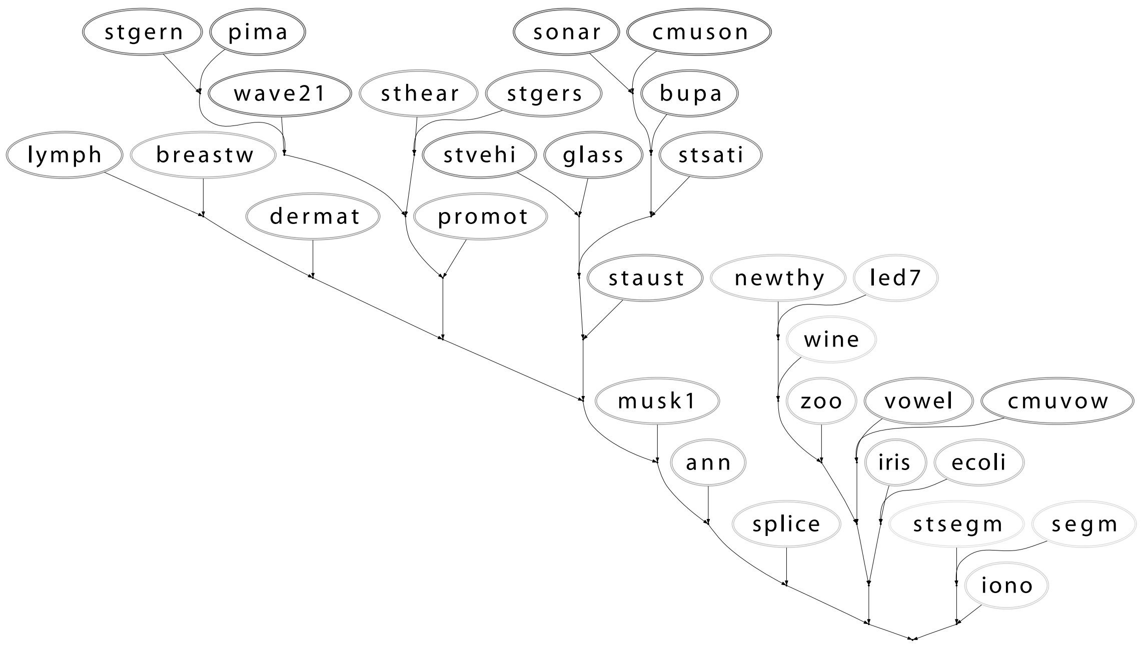

Fig. 3. Clustering of learning tasks using Average Link Agglomerative Clustering.

of Irvine MLDB tasks selected, there exists a relatively dense clustering of tasks about a well-defined center. If this result generalizes to learning tasks in general, it suggests that learning algorithms which focus on this part of the task space will tend to improve performance on a large number of learning tasks. Conversely, algorithms focused on performance on the outlying tasks will tend to cover fewer learning tasks unless they have a broad coverage of the task space in these outlying areas.

To verify the robustness of this approach, other clustering techniques were also applied to these tasks. These clustering methods included minimum link agglomerative clustering and recursive partitioning using maximum graph cut. The observations found in these clusterings were similar to those made above with average link agglomerative clustering.

In general, the choice of which clustering technique to employ may be influenced by the intended use of the clusters. If, for example, clusters are being used to decide which tasks may be used for transfer learning, agglomerative clustering would be more appropriate than graph bisection. In this case, clustering is used to gain general insight about the space of learning tasks. Each clustering technique may offer a different insight. However, it is encouraging that the observed clusters appear to be rather robust regardless of the clustering technique used.

5. Conclusion and future work

This research provides a method for computing the distances between learning tasks. This is done by building on the concept of an algorithm distance and reversing it to allow for distances between tasks. This research gives examples of visualizations for use in understanding the relationships between learning tasks. It was observed that the apparent dimensionality of the space of learning tasks (as measured using the DICES metric) is significantly lower than the potential dimensionality. The task arrangement for the thirty tasks observed exhibited only two strong dimensional components (and about a dozen weaker components). Learning tasks were observed to form cliques representing clusters of tasks that are similar to one another.

It is observed that learning tasks which differ from each other only in the type of encoding chosen for the input features exhibit a nonneglible DICES distance between each other, although this distance is still significantly lower than the general DICES distance between two arbitrary tasks. This suggests that learning algorithm behavior is somewhat sensitive to implementation choices such as encoding strategy. Conversely, this also provides evidence that the DICES distance is an effective tool for finding tasks that are closely related to one another.

Future work in this area includes research into uniqueness functions that can be used to weight data sets for empirical results. With the ability to generate a distance matrix, this work may also be applied to transfer learning.

Some work has been done on using distance relationships between tasks to guide task selection for presenting empirical results across a broad range of tasks while minimizing redundancy in tasks (that is, tasks which are closely related). Work is also being done in producing weightings for empirical results which can be applied to already published research in order to determine the broadness of the algorithm(s) presented by them.

Other future directions for this research include examining task space from the perspective of algorithm coverage and determining where individual learning algorithms perform well or poorly in relation to tasks and to each other. It is hoped that in being able to visualize and understand learning task space better, research might be made more effective at targeting unexplored techniques for machine learning.

References

[1] D.W. Aha, Generalizing from case studies: A case study. In *Proceedings of the Ninth International Conference on Machine Learning*, ICML'2000, Aberdeen, Scotland, Morgan Kaufmann, 1992, page 1–10.

[2] J. Baxter, Learning model bias, in: *Advances in Neural Information Processing Systems*, D.S. Touretzky, M.C. Mozer and M.E. Hasselmo, eds, volume 8, The MIT Press, 1996, pp. 169–175.

[3] C.E. Brodley, Dynamic automatic model selection. Technical Report UMCS-1992-030, University of Massachusetts, February 1992.

[4] D. Cohn, L. Atlas and R. Ladner, Improving generalization with active learning, *Machine Learning* **15**(2) (1992), 201–221.

[5] C.J. Merz and P.M. Murphy, UCI repository of machine learning databases, Available at http://www.ics.uci.edu/∼mlearn/MLRepository.html, 1996.

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

[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*, ICMLfl2000, Morgan Kaufmann, San Francisco, California, 2000, pages 743–750.

[8] S. Thrun and J. O'Sullivan, *Discovering Structure in Multiple Learning Tasks: The TC Algorithm*, In International Conference on Machine Learning, 1996, pages 489–497.

[9] S.B. Thrun, Is learning the n-th thing any easier than learning the first? in: *Advances in Neural Information Processing Systems*, D.S. Touretzky, M.C. Mozer and M.E. Hasselmo, eds, volume 8, The MIT Press, 1996, pp. 640–646.

[10] S.B. Thrun and T.M. Mitchell, Learning one more thing, In IJCAI, 1995, pages 1217–1225.

[11] D.H. Wolpert and W.G. Macready, No free lunch theorems for search. Technical Report SFI-TR-95-02-010, Santa Fe Institute, Santa Fe, New Mexico, 1995.