EnsembleMatrix: Interactive Visualization to Support Machine Learning with Multiple Classifiers

Justin Talbot*†, Bongshin Lee†, Ashish Kapoor†, Desney S. Tan†
*Stanford University
353 Serra Street, Stanford, CA 94305
jtalbot@stanford.edu

†Microsoft Research
One Microsoft Way, Redmond, WA 98052
{bongshin, akapoor, desney}@microsoft.com

ABSTRACT
Machine learning is an increasingly used computational tool within human-computer interaction research. While most researchers currently utilize an iterative approach to refining classifier models and performance, we propose that ensemble classification techniques may be a viable and even preferable alternative. In ensemble learning, algorithms combine multiple classifiers to build one that is superior to its components. In this paper, we present EnsembleMatrix, an interactive visualization system that presents a graphical view of confusion matrices to help users understand relative merits of various classifiers. EnsembleMatrix allows users to directly interact with the visualizations in order to explore and build combination models. We evaluate the efficacy of the system and the approach in a user study. Results show that users are able to quickly combine multiple classifiers operating on multiple feature sets to produce an ensemble classifier with accuracy that approaches best-reported performance classifying images in the CalTech-101 dataset.

Author Keywords
Visualization, interactive machine learning, ensemble classifiers, object recognition, Caltech-101.

ACM Classification Keywords
H.5.2 [User Interfaces]: Graphical User Interface; I.2.6 [Learning].

INTRODUCTION
Machine learning (ML) techniques utilize computational and statistical methods to automatically extract information from data, thus allowing computers to "learn." Recently, human-computer interaction (HCI) researchers have taken increasing interest in building classifiers using machine learning for their applied value. For example, using applied machine learning, HCI researchers can disambiguate and interpret noisy streams of data and develop novel input modalities, analyze complex patterns in data to perform predictions or diagnoses, or infer user intent to optimally adapt interfaces to assist users.

As machine learning becomes more widely applied, both within the HCI community and beyond, a critical human-computer interaction challenge is to provide adequate tools to allow non-experts to wield ML techniques effectively. In order to design these tools, we must start at current best practices in applied machine learning and identify tasks that can be supported or augmented by an effective combination of human intuition and input with machine processing.

In this paper, we restrict our attention to multiclass classification problems. For such problems, the goal is to develop an algorithm ("classifier" or "model") that will assign input data to one of a discrete number of classes. Standard classification considers problems where there are two possible classes. Multiclass classification permits any number of classes (consider classifying handwritten characters as one of the 26 letters of the alphabet) and is generally considered a much more challenging problem in ML.

At present, a common applied machine learning workflow is to iteratively develop classifiers by refining low-level choices such as feature selection, algorithm selection, parameter tuning, and so on [27]. This is usually performed using manual trial and error in an approach that represents hill-climbing through the model space. Models are compared using accuracy or other coarse summaries. These simple summaries provide little direction on how to improve the classifier. This problem is exacerbated in multiclass problems where single value summaries, such as accuracy, can be very misleading. To address this issue, previous research has focused on developing tools for better explaining a single classifier's behavior (or misbehavior).

We assert that this workflow is often sub-optimal. It obscures the beneficial dependencies and complementarities that exist between classifiers and discards the context of the space of possible models. It can and does often lead to poor local maxima and the large work in generating the series of models is often wasted as there are no means to compare and evaluate the trajectory of models being learned.

In our work, we explore a radically different workflow. In the proposed workflow, we instead place emphasis on taking a holistic perspective provided by a set of models selected from the model space. By supplying a visual sum-

CHI 2009, April 4–9, 2009, Boston, MA, USA.

mary that spans multiple classifiers, we help users understand the models' various complimentary properties. This experience and other insights gained in this process can provide users with ideas and methods for combining the multiple classifiers in order to build a model that is vastly superior to any of its components. The individual classifiers could be a product of the user's own explorations, for example using a tool such as Weka [14] to generate multiple models. Or they could be classifiers generated by other researchers working on similar problems.

In this paper, we present EnsembleMatrix, a system that provides visualization of core machine learning elements such as the confusion matrix in order to support greater understanding of relationships between individual models. It also implements interactive ensemble model combination strategies so that users can directly manipulate the visualizations in order to build combination multiclass classifiers.

Specifically the main contributions of this paper are:

1. Presenting EnsembleMatrix, a system that leverages interactive visualization to allow users to browse and learn properties of classifiers by comparison and contrast;
2. Describing interactions in EnsembleMatrix that allow users to easily create an ensemble classification system by discovering appropriate combination strategies;
3. Presenting results of a user study we conducted with the CalTech-101 image classification task [11]. Results showed that users were not only able to use the system, but that they were able to quickly create classifiers that rival state of the art in this problem space.

BACKGROUND AND RELATED WORK

Applied Machine Learning

Machine learning, in particular classification, has become an increasingly important tool in HCI research, and more generally in the development of modern software. In a recent survey of 112 HCI professionals, about one third responded as having used machine learning in their HCI work [25]. For example, using applied machine learning, HCI researchers have worked on disambiguating and interpreting noisy streams of data to develop novel input modalities. This includes work with interpreting explicit intent such as in handwriting and speech recognition as well as muscle-computer interfaces [30]. But it is also often used to detect implicit states, such as when using physiological signals to infer affect [28], or brain-computer interfaces to infer cognitive state [16]. Another domain of work that has extensively leveraged machine learning techniques is in building adaptive interfaces. For example, one sub-area that has been worked on is in modeling human behavior to predict things like interruptibility [12, 19].

In their work, Fails and Olsen describe the importance of human involvement to provide training data and propose an interactive machine learning model that allows users to train, classify, and correct classifications in a continuously iterative loop [10]. In our work, we apply the same basic philosophy to the creation of models rather than focusing on training existing models.

In a recent study, Patel *et al.* evaluated the current use of machine learning by non-expert researchers and identified three difficulties: difficulty in applying an iterative exploration process, difficulty in understanding the machine learning models, and difficulty in evaluating performance [27]. They suggest creating a library of models known to work well for a variety of common problems. Such a library could be the source of an initial set of models. In our work, we address some of the issues raised in that paper and present a methodology that could leverage and extend the library of models that they propose.

Ensemble Machine Learning Models

Many researchers have looked into the general problem of combining decisions from multiple classifiers. Simple rules such as majority vote, sum, product, maximum and minimum of the classifier outputs have been popular and often produce results better than individual classification system [21]. One problem with these fixed rules is that, it is difficult to predict which rule would perform best. On the other spectrum lie critic-driven approaches, such as layered HMMs [26], where the goal is to "learn" a good combination scheme using a hierarchy of classifiers. The disadvantage with these methods is that they require a large amount of labeled training data, often prohibitive for HCI work.

Large sets of classifiers can be automatically generated using popular methods such as Boosting [31] and Bagging [5]. Also related to ensemble learning is "feature level fusion," in which features are first fused and then a machine learning algorithm is used to learn a classifier on top of these features [34]. However, the majority of the work in this space is aimed at learning 2-class classifiers. While strategies exist to extend 2-class classifiers to multiclass problems (such as one-vs-all or pair-wise classification), the combination is relatively naïve and does not take advantage of any semantic or functional structure that might be present in the multiclass classification problem.

Visualization for Machine Learning

We instantiate our work in an interactive visualization system that allows users to explore data and classifiers, and then to combine them into a single classifier. To aid machine learning development, researchers have explored visualizing specific machine learning algorithms, including naïve-Bayes [2], decision trees [1], SVMs [6], and HMMs [7]. A previous study by Ware *et al*. has shown that such tools can produce better classifiers than automatic techniques [36]. Since these visualizations and interaction techniques are tied to specific algorithms, they do not support comparisons across algorithm types.

More general techniques that apply across algorithm types include ROC and Cost curves [8], which support evaluation of model performance as a function of misclassification

costs. These visualizations are commonly used in the machine learning community. But, for practical purposes, they are restricted to binary classification tasks and they do not directly support iterative improvement of a classifier.

One of the most frequent visualization techniques in machine learning is to plot the data instances in some projection of feature space and visualize model prediction boundaries [13, 29]. This approach generalizes across algorithms at the expense of not showing the internal operation of the particular algorithm. Urbanek [33] designed visualizations of summary statistics derived from a large set of decision trees generated by boosting and Stiglic *et al.* [32] proposed a method for visually comparing a small set of decision trees produced by boosting.

Instead of visualizing the data space with classification boundaries, our work focuses on visualizing recognition results and summary statistics, specifically those found in the confusion matrix. We assert that the recognition structure of a particular algorithm enables users to see what changes have to be made to improve the model.

Bertin, who first introduced matrix visualization to represent networks, showed that matrices can be used to exhibit high-level structures by reordering the rows and columns of the matrix [3]. Following this, many researchers have been trying to exploit the benefits of matrix reordering. For example, ConSet provides an overview using an improved permutation matrix to enable users to easily identify relationships among sets with a large number of elements [20]. MatrixExplorer employed reordered matrix-based representation in addition to node-link diagrams to support social network analysis [17]. Wang *et al.* used shaded similarity matrix with reordering to visualize clusters in classification [35]. But, they designed the system for visualizing similarity of data instances, not confusion matrices. We use similar techniques to reorder the confusion matrices in an effort to provide visual insight to users.

ENSEMBLEMATRIX

In this section we describe EnsembleMatrix, a system that helps machine learning practitioners explore and combine individual classifiers into ensemble classifiers.

Visualizing the Confusion Matrix

The EnsembleMatrix interface consists of three basic sections: the Component Classifier view on the lower right, which contains an entry for each classifier that the user has imported to explore, the Linear Combination widget on the upper right, and the main Ensemble Classifier view on the left. See Figure 1.

Both the Component Classifier and the Ensemble Classifier views visually represent a classifier as a graphical heat-map of its confusion matrix. A confusion matrix represents classification results by plotting data instances in a grid where the column is an instance's predicted class and the row an instance's true class. Confusion matrices can reveal trends and patterns in the classification results (Figure 2) and by extension reveal behavior of the classifier itself. We selected the confusion matrix as the core visual element of our system since it can represent results from all classification algorithms and interpretation is relatively algorithm-agnostic. This nicely complements previous work that has focused on visualization properties of specific algorithms.

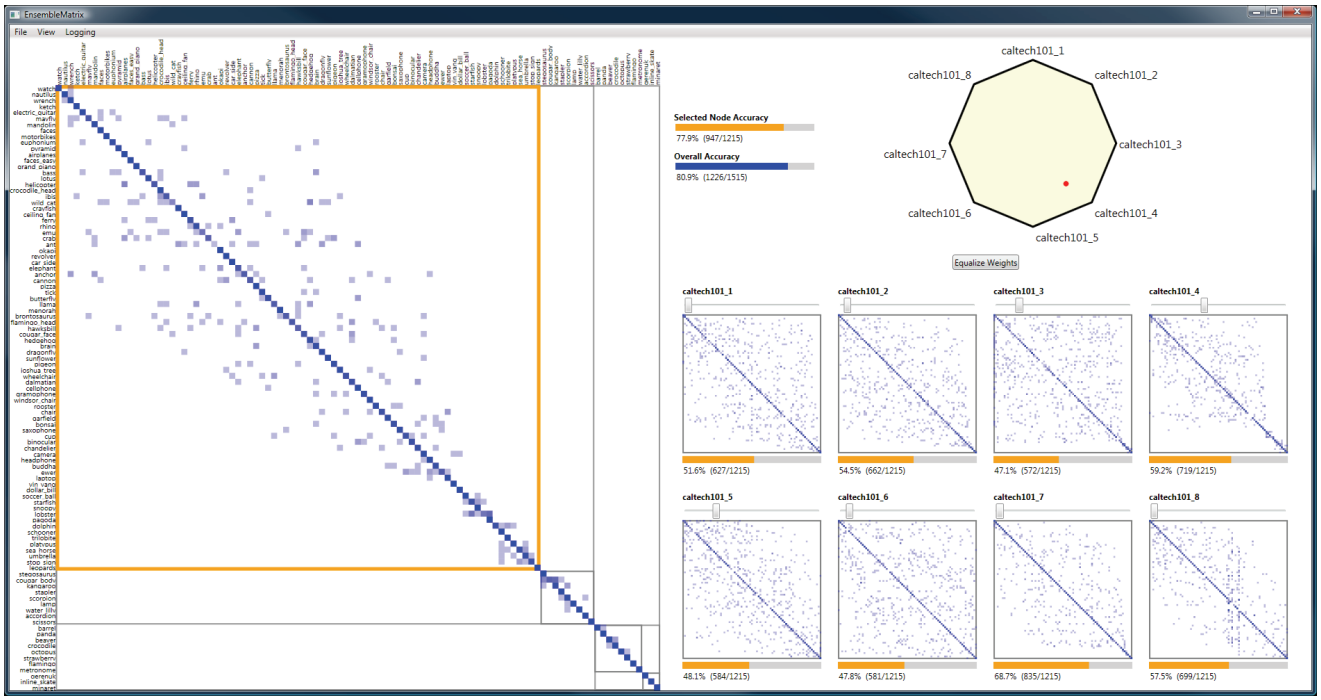

Figure 1. Primary view in EnsembleMatrix. Confusion matrices of component classifiers are shown in thumbnails on the right. The matrix on the left shows the confusion matrix of the current ensemble classifier built by the user.

	1	2	3	4	5	6	7	8	9	0
1	91	0	1	2	0	1	3	1	0	1
2	1	89	1	1	1	1	2	1	2	1
3	1	2	48	1	2	0	3	40	1	2
4	2	0	0	83	0	3	7	2	3	0
5	10	7	12	10	30	4	5	11	5	6
6	0	1	1	1	1	95	0	0	1	0
7	2	0	1	0	1	1	94	0	1	0
8	1	2	47	1	1	1	2	43	0	2
9	4	0	0	0	1	0	0	3	92	0
0	2	1	1	1	2	1	2	1	2	87

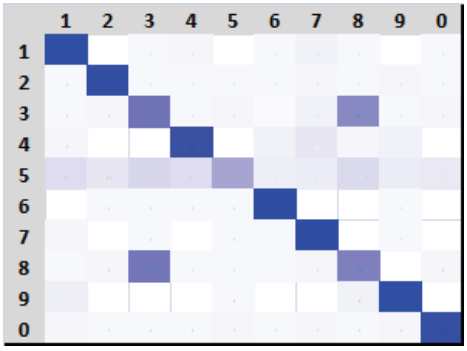

Figure 2. Representations of confusion matrix for a handwritten digit classification task. (top) standard confusion matrix; (bottom) heat-map confusion matrix. It is much easier to identify underlying patterns in the visual representation; 3 and 8 are often misclassified as each other and 5 is misclassified as many different numbers.

As with other matrix visualizations, the ordering of the matrix can greatly influence the patterns visible. EnsembleMatrix orders each of the Component Classifier matrices independently to highlight sets of classes which are frequently confused by that particular classifier. This corresponds to grouping clusters in an adjacency matrix. Additionally, as users update the Ensemble Classifier view, the main matrix is reordered interactively. This necessitates a fast reordering algorithm, so we chose to use the barycenter heuristic [23], borrowed from the layout of bipartite graphs.

Small horizontal bars below each matrix in the Component Classifier view show the accuracy for each of these classifiers. A similar set of bars in the Linear Combination pane show the overall accuracy of the current Ensemble Classifier and the accuracy of the currently selected partition.

Interacting with EnsembleMatrix

EnsembleMatrix provides two basic mechanisms for users to explore combinations of the classifiers. The first is a partitioning operation, which divides the class space into multiple partitions. The second is arbitrary linear combinations of the Component Classifiers for each of these partitions.

We selected these two operations since they are the core portions of a variety of more complicated machine learning algorithms. We expect that the most effective use of these operations is in tight conjunction, iterating back and forth, with each other. For example, one possible strategy is to adjust weights and find good partitions, and then to recursively iterate to the sub-partitions.

To enable these mechanisms, we assume that our component classifiers can produce a vector of real numbers, one number per possible class. The predicted class is the class with the maximum numeric value. This formulation is very natural in many multi-class classifiers including SVMs and logistic regression. If a classifier can only output a single predicted class, this can be represented as a vector with a single non-zero value.

We also suggest that the component classifiers be trained in a robust manner, such as with 10-fold cross validation. Doing so increases the likelihood that partitioning or linear combination operations which improve accuracy on the training data will generalize well to the entire dataset [9].

Partitioning

Partitioning the class space separates the data instances into two subsets, allowing the user to develop classifiers specialized for each subset. To specify a partition, the user selects a vertical line on a matrix. Data instances which fall to the left of the partition are placed in one set and those which fall on the right in the other. Once partitioned, data instances may not cross the partition line despite further partitioning or refinements to the classifiers on either side. Technically, this means that the two subset classifiers are restricted to only predicting classes which fall on the appropriate side of the partition line. This is accomplished by maximizing over an appropriate subset of the numeric vector. This constraint makes the behavior of the partition easier to interpret and to represent visually.

One impact of this partitioning constraint is that data instances that are on the wrong side of the partition will remain on the wrong side despite further refinement. To emphasize this fact, EnsembleMatrix automatically adds a horizontal split at the same position as the vertical, thus creating four quadrants for each partition (Figure 1). Data instances in the off-diagonal quadrants are those which will remain incorrectly classified.

By clicking on an on-diagonal quadrant users can select either the left or right subset and can continue refining just the classifier for that subset by further partitioning or by adjusting the linear combination. The selected subset is highlighted with an orange border and the Component Classifier thumbnail views update to only show data instances contained on that side of the partition (Figure 3).

In practice, we found that over-partitioning the space led to poor generalization since the sample of data items was too small to adequately represent the whole dataset, but that users seemed able to gauge when they should stop.

Linear Combinations

There are two ways in which users can manipulate the linear combination of Component Classifiers. First, EnsembleMatrix provides a simple 2D interpolation control, the Linear Combination Widget in the upper right (Figure 1).

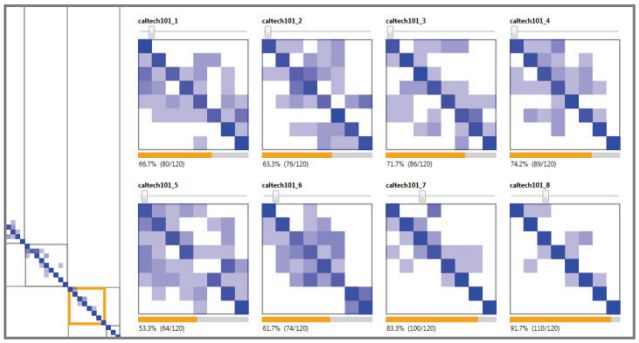

Figure 3. After partitioning the matrix, selecting a partition, outlined in orange, causes the thumbnails to display only the data instances in that partition. The component classifiers demonstrate very different behavior in this partition, including clustering and large differences in accuracy.

The user can interactively scrub inside this polygon to specify classifier weights. To make the appropriate mapping of position to weight, we parameterized the polygon using Wachspress coordinates [24]. Wachspress coordinates have two important properties: at the vertices of the polygon the weight is one for the classifier at that vertex and zero for all other classifier, and at the center the classifiers all have the same weight. These points seem to be the most natural in this space and we wanted to ensure that the subspace represented by the polygon included those points.

This subspace is only 2-dimensional and hence much of the weight space is not included in the polygon. Thus, we also provide individual sliders under each Component Classifier to allow the user to specify an arbitrary normalized combination of weights should they need the extra control.

Given the user specified weights, EnsembleMatrix simply takes the vectors of numeric values output by the component classifiers and linearly combines them. The class with the maximum value is chosen as the predicted class.

Partitioning and reweighting can be done an arbitrary number of times and in any order, leading to a large number of possible refinement strategies.

USER STUDY

We conducted a formative user study to examine the usability of EnsembleMatrix and the efficacy with which users could use the system to explore component classifiers and create ensemble ones. To do this, we used the CalTech-101 dataset [11] to perform an image recognition benchmark task. This task is an active problem in machine learning and has been used to develop numerous new algorithms. The dataset contains 3030 images grouped into 101 classes, and the task is to build a classifier that categorizes them. This problem was selected since it is a challenging unsolved problem in machine learning, allowing us to compare EnsembleMatrix with the best-known ML techniques.

We collected eight classifiers developed by other researchers that have been tested on the CalTech dataset. These are some of the best single feature-type classifiers currently available for this problem. Our goal was to learn if applied machine learning practitioners could use EnsembleMatrix to discover interesting properties of the eight classifiers and more importantly if they could improve accuracy by effectively combining the classifiers using the system.

Methodology

We recruited seven participants (including one pilot) from a software company via an internal mailing list for those interested in machine learning. Except for the single female pilot, all were male ranging in age from 26 to 34. Two of the participants were machine learning researchers with over 8 years of experience, one was a developer using machine learning for a shipping product, and the remaining three had exposure with machine learning only through undergraduate and graduate coursework.

We split the study into two parts. The first part was a semi-structured interview designed to elicit a range of possible strategies for solving this problem. We asked participants to describe potential approaches they would try to solve this problem using their current knowledge, practices, and tools. We provided participants with basic information about the eight classifiers and asked them to imagine that their goal was to develop a new, more accurate, classifier. We also provided additional information about the dataset and about the classifiers when asked by the participant.

If a participant described approaches that did not involve combining the classifiers, we asked follow-up questions specifically asking them to consider how they might combine the classifiers. We also asked participants to estimate how long it would take them to reach a point where they would be able to predict with high confidence whether or not their approach would work. This part of the study lasted approximately 10 minutes.

In the second part, we trained participants to use EnsembleMatrix, using data from a handwritten digit classification task. This classic task had participants explore 4 different classifiers in an effort to build an ensemble classifier that recognized the images of handwritten digits. This problem is easy to explain, since people quickly understand handwriting recognition, and allowed us to focus on teaching them how to use the interface. Also since the problem only has 10 classes (the digits 0 through 9) it makes it easy for participants to gain insights and manipulate the classifiers easily. During this phase, we did not explicitly discuss or teach participants any particular exploration strategy.

In the test task, we asked the participants to use EnsembleMatrix to explore the eight CalTech-101 classifiers and to build a combined classifier with the highest possible accuracy. Participants were encouraged to think aloud as they worked. This portion lasted 30-45 minutes and ended when the participant thought they could no longer improve the ensemble classifier.

Ensemble Classifier Details

In this section, we describe the processing we used in order to create the Ensemble Classifier.

Classifying object categories in images is a hard multi-class problem (101 classes for Caltech-101). Our experiments in this paper use 30 images per class (3030 images in total), and are exactly same as the ones used by Varma and Ray [34]. We divided the dataset equally into train and test sets (i.e., 15 images from each class in train and test sets). The training set was used to train the component classifiers with 10-fold cross validation and results of this are presented to the user in EnsembleMatrix. The user never interacts with test data, which is used to compute the reported accuracy.

As component classifiers, we used 8 Gaussian Process regression [18] classifiers trained on different features. These features, developed independently by other computer vision researchers, characterize object shape and appearance in an image. Specifically, we looked at the following eight kinds of features: Dense Pyramid Match Kernel (PMK) [15], Spatial PMK [22], AppColour, AppGray, Shape 180, Shape 360 [4], Geometric Blur (GB), and Geometric Blur with distortion [37]. Each of these features exploits different characterization of images (for example, color, edges, etc.). While accuracies of these classifiers are comparable, they each have individual strengths and operate on different parts of the class space with varying accuracy.

Results: Semi-Structured Interview to Elicit Approaches

In our discussion, participants suggested multiple possible approaches to solving the classification problem. The approaches can be broken into two general areas. First, participants suggested working with the underlying features. Two suggested training a new classifier on the combined feature sets of all 8 classifiers, and three suggested looking at misclassifications in individual classes to help select additional features. Second, participants described schemes which combine the component classifiers. Combining the classifiers with stacking was suggested three times, with boosting three times, and with majority voting twice.

Implicitly, all participants assumed that they had or could derive useful understanding of the data and the existing classifiers in order to improve upon them. Two participants (1 expert) explicitly mentioned that they would explore the classifiers to see if they are complementary, either by looking at the confusion matrices or by plotting data instances against number of times classified correctly. One participant suggested combining semantically related classes to reduce the number of classes and to increase accuracy.

Participants' estimates of time required to convince themselves that their approach would work varied. Both machine learning experts gave estimates of less than one hour. One participant estimated 1 day using the Weka tool. Two participants estimated more than 2 weeks, including implementation time. One participant did not give a time estimate.

Time estimates seemed to vary inversely with machine learning experience as might be expected. We hypothesize that the short times for the machine learning experts were largely due to the fact that they already had written code for the approach they thought they would use. The long times for the non-experts reflects both time to code as well as built-in uncertainty in the effectiveness of their approach.

We assert that the two week turnaround projected by non-experts is optimistic and problems of this nature usually take much longer to fully explore. Regardless, we assert that even two weeks to test a single hypothesis would prohibit rapid exploration. While the machine learning experts seemed relatively confident of the tools that they had, they acknowledged a lack of support for data exploration and higher level tools for applied machine learning problems.

Results: Using EnsembleMatrix to Build Classifiers

Observations: Making sense of the individual classifiers

We observed participants as they were performing the ensemble classifier construction task in EnsembleMatrix. In general, participants tried a large number of strategies. Most participants utilized the tool and specifically the confusion matrices to understand the current state. They would look at individual confusion matrices trying to understand the behavior of a classifier in isolation. Some then noticed confusion of class clusters under some of the classifiers and tried to determine why these classes were confused. They did this by trying to intuit how the features used by a classifier led to particular class confusions or by looking across classifiers to find semantic relationships discovered by the classifiers. They would also examine individual classes trying to understand the behavior of data instances in that class across multiple classifiers to gain insights.

Using these strategies, participants independently made various discoveries. A majority of participants immediately concluded that errors were evenly distributed in some classifiers, e.g. individual class accuracy is pretty even. This is not surprising, as the graphical representation of the confusion matrix makes this very apparent. They also found that some classifiers produced clear confusion clusters, and that those clusters seemed to be different in other classifiers. Those who had previously used combination classifiers recognized that this implied that the classifiers were complementary. However, upon inspection of the classes, few participants could directly infer the semantic meaning of the clusters. This is not unusual since the features used for the classifiers do not usually correspond well to perceptual features as picked up by humans. On the whole, they seemed to be able to stratify the classes into ones that were easy to classify and ones that were easily confusable, which they claimed would allow them to focus their efforts in improving the classifiers.

Observations: Building an Ensemble Classifier

To build the combination classifier, all participants changed the weights frequently to see changes in accuracy. Two participants (1 expert) used the sliders almost exclusively,

claiming they wanted fine control over the weight combinations. They performed only slightly better than the other participants. Participants also used partitioning in slightly different ways. For example, one participant used the partitioning mechanism to remove portions of the matrix that were already perfectly classified, and then refined the weighting scheme on the remainder. Another partitioned the matrix repeatedly and then visually compared the performance of the classifiers within just one leaf node at a time.

While experimenting with this problem ourselves, we discovered a relatively simple strategy to improve accuracy. We used the matrix reordering to find strong confusion clusters in individual classifiers and recursively isolate the clusters. We then used linear combination to optimize the individual clusters in the leaf nodes. This algorithm led to high accuracy and good generalization. It is encouraging that participants seemed to independently find similar, even if not identical, strategies.

On the whole, participants were able to use the linear combination tools to find accuracy peaks in weighting space that were relatively robust to changes in weights, and were able to construct classifiers that performed much better than the individual component classifiers. They also found that hierarchical partitioning between confused clusters, combined with linear combination, produces better accuracy than linear combination alone.

Interestingly, some participants expressed at various points in the experiment feeling that they had not learned a lot about the classifiers or the dataset. However as already discussed, many in fact had interesting insights and were able to relatively quickly create a classifier that performed very well. We believe that this disconnect was due in part to participants' expectations that they would be able to explore the behavior of individual classifiers and understand specific reasons for misclassification. As discussed in the introduction, our goal with EnsembleMatrix was to support higher-level reasoning across classifiers. We believe that the two approaches are not mutually exclusive, and would like to integrate lower-level classifier explorations in future versions of the system.

As an additional point, users requested the ability to undo operations and to save bookmarks of promising areas to explore further. This was not surprising and we will include these features in the next version of EnsembleMatrix.

Quality of the Ensemble Classifiers

After allowing participants to explore independently for about 15 minutes, we asked them to try to use the linear combination and splitting tools to build a classifier with maximum accuracy on a hold out test set. One participant chose not to perform this task. The other five participants made varying levels of effort to find the maximum, using 10-30 minutes. Results are plotted in Figure 4.

These results show that participants were able to build a combination model that attained relatively high classification accuracies. The best individual classifiers (some of which were used as component classifiers in this study) achieve about 65% accuracy, on average. The best combination classifiers handcrafted by experts and published within the last year, achieve around 87.82% [34]. Although restricted to short period of time, our five participants accomplished 87.8% on average (86.5% minimum, 89% maximum), which almost ties with the highest published accuracy. The small range implies that participants' performance was consistent.

Furthermore, three participants outperformed the highest published accuracy. The main reason behind the excellent classification performance is the fact that a user is now able to create a hierarchical classification system that respects and take advantages of structural information amongst various classes. Most of the earlier approaches that were applied to Caltech-101 limited themselves to simple single

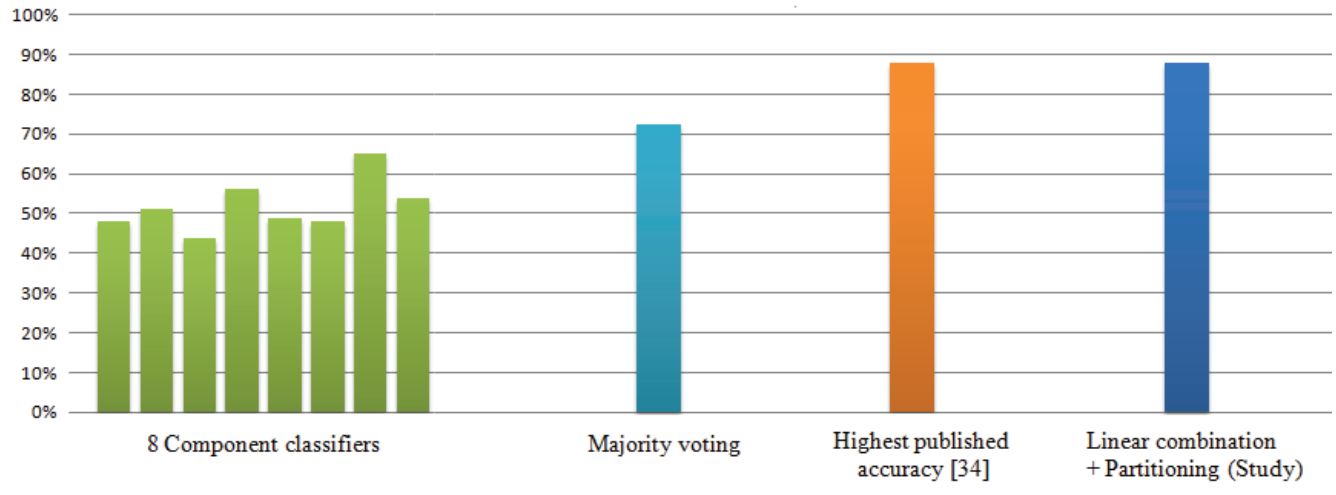

Figure 4. Average accuracy achieved in EnsembleMatrix on a hold out test set by 5 participants in our study compared to other approaches.

models that did not exploit any hierarchy. Further, the individual base classifiers are some of the best methods in this space; consequently the ensemble consistently achieves high classification accuracy. In a parallel thread of work, we are now exploring this approach for a variety of other problems, and will report on results in future work.

Interestingly, one of the participants who attained the highest accuracy on the training set built a model that did not work quite as well on the test set. This was due to the fact that he over-partitioned the space of classes and overfit the data. Even so, he was able to do relatively well and immediately recognized the reason for his lower performance on the test set. In future work, we would like to explore computational as well as interface techniques that might prevent this behavior.

After completing this task we asked the participants to evaluate how close they thought their result was to the best possible with this dataset and classifiers. Four participants were able to make predictions which were very close (±1%) to the best possible results published in the literature.

As a post hoc analysis, we evaluated how well some of the techniques suggested by participants in the interviews would work (see Figure 4). Majority voting does relatively poorly, only reaching a 72% accuracy rate. According to the literature the best results from combination of different methods using SVMs is 87.82% [34]. However, this work is very new and it is unlikely that a non-expert user would be aware of this technique. Merely combining the underlying features and training with a relatively simple classification algorithm is likely to perform rather poorly. As discussed in the related work section, there are many possible ways to combine classifiers with machine learning. Most appealing to a majority of researchers is boosting; however, presence of a large number of multiple classes, together with Gaussian Process-based classifiers makes application of boosting non-trivial. One plausible way of applying boosting is to consider the outputs of the base level classifiers as features and then apply a boosting algorithm such as AdaBoost. However, this achieves only a 73% accuracy rate and highlights challenges in assuming that existing combination strategies can be used as a blackbox. There are many other boosting or stacking strategies that can be tried and may work better. However, the time necessary to select, implement, and test these strategies would be significantly longer than the 30-45 minutes required by EnsembleMatrix participants in our study.

We also explored using EnsembleMatrix's linear combination and partitioning tools independently. Partitioning alone performs poorly. When using the linear combination tool alone, we were able to achieve an accuracy of 87.3%. Comparing this to the maximum accuracy achieved in our user study, 89%, shows that both tools are useful.

DISCUSSION

The user study showed that users could use EnsembleMatrix to effectively explore the data and create ensemble classifiers that performed much better than any of the given individual classifiers. This is encouraging and suggests that the system and the approach are viable and worth further exploration. In this section we discuss the two most interesting issues, and opportunities, that came up in the course of this work.

Confusion from Reordering Classes within Matrices

As previously noted, each of the matrices is reordered independently in order to highlight structure in each matrix. At times this was confusing to users who tried to visually compare two matrices and mistakenly assumed that the ordering was the same. This was an especially large issue for the one user had split the matrices down to small submatrices and tried to perform visual comparison of individual rows or columns. The reordering also caused the main matrix to visually jump around as the user changed the weighting. This was distracting and made visual comparisons between matrices difficult. We believe that it also made it hard to derive semantic insight into the classes since they were ordered differently in each classifier.

In choosing ordering we are faced with three contradictory requirements. First, each matrix should be ordered to show the confusion structure produced by a single classifier. Second, the ordering should be consistent across matrices to support visual comparison. Third, users often have a semantic organization of classes that they would like to have respected by the ordering. Finding a visualization that conforms to all three constraints is an area for further research.

Why Human in the Loop rather than Automating it all?

A concern expressed by a number of our user study participants is why we would not just automate much of the exploration. In retrospect, it is now clear to us that some parts of EnsembleMatrix could probably be automated. For example, once a human has found a particular partitioning strategy and can well-define the strategy, there is no reason for that human to have to mechanically apply it to all explorations and all datasets. Similarly, searching for the absolute maximum accuracy weighting combination is something that may be better supported by automation.

However, even these straightforward examples would be non-trivial to automate. First, exhaustively exploring the space of models, especially with non-trivial number of base classifiers is usually going to prove to be computationally intractable. Hence, the user will still have to specify what subspace should be explored. Second, for purposes of generalization, the user may prefer a local maximum in a broad high accuracy region, to the global maximum in a relatively narrow peak. Automating the correct decision boundary between these two choices is not straightforward. However, in EnsembleMatrix, a user can (and users in our study did) scrub through the polygon space to get a feel for peak robustness and make an intuitive decision.

Also, we doubt that any given ensemble construction strategy would work for all datasets and classifiers. Future work would include exploring what other combination strategies

arise in other datasets from the weighting and partitioning mechanisms we have provided. Additionally, we want to explore what other exploration mechanisms are useful. For instance, permitting the user to split correctly classified data instances from incorrectly classified instances would allow a boosting-like strategy to be employed. We hope to find a small set of mechanisms which are:

1. Simple to understand and to represent visually
2. Represent the core operation from a range of machine learning ensemble approaches (if a user finds that the operation works in EnsembleMatrix, then the user could explore using more complicated ML approaches in other tools)
3. Composable (various existing machine learning techniques and many novel combinations of strategies should be expressible with the mechanisms)

Finally, it is important to note that this task certainly did not appear automatable before using the system. From their responses in the semi-structured interviews, it is not clear that any of our study participants would have independently formulated the solutions they found using EnsembleMatrix. We consider the fact that the tool permitted non-expert users to quickly explore and discover a novel ensemble-creation strategy to be a measure of the success of the system. The fact that the strategy can now possibly be automated for this dataset is an additional bonus.

CONCLUSION

We presented EnsembleMatrix, an interactive visualization system for exploring the space of combinations of classifiers. We demonstrated that EnsembleMatrix can be successfully used to explore and develop classifiers for an important problem in machine learning, and argued that the basic approach would generalize well to other problems. While EnsembleMatrix is just a first step, we believe that including the human at the stage of combining machine learning classifiers can be a more effective use of human knowledge, judgment, and time than at the now-standard approach of tuning individual models.

It is particularly exciting to us that EnsembleMatrix has allowed us to discover techniques that outperform state-of-the-art in classifying a benchmark computer vision and machine learning problem using the CalTech-101 dataset. Further testing and documenting those results remains future work. In other future work, we hope to resolve many of the problems we have noted with EnsembleMatrix and explore its application to a variety of datasets and problems within human-computer interaction research.

ACKNOWLEDGMENTS

The authors would like to thank George Robertson and Dan Morris. Additionally, we thank the following for providing kernel matrices: Alex Berg, Anna Bosch, Kristen Grauman, Jitendra Malik and Andrew Zisserman.

REFERENCES

1. Ankerst, M., Elsen, C., Ester, M. and Kriegel, H.-P. Visual classification: an interactive approach to decision tree construction. *Proc. KDD 1999*, (1999), 392-396.
2. Becker, B., Kohavi, R. and Sommerfield, D. Visualizing the Simple Bayesian Classifier. [ed.] Fayyad, U., Grinstein, G. and Wierse, A. (2001), 237-249.
3. Bertin, J. Semiology of graphics. University of Wisconsin Press, Madison, WI, USA,1984.
4. Bosch, A., Zisserman, A. and Munoz, X. Representing shape with a spatial pyramid kernel. *Proc. CIVR 2007*, (2007), 401-408.
5. Breiman, L. Bagging predictors. *Machine Learning* (1996), 123-140.
6. Caragea, D., Cook, D. and Honavar, V.G. Gaining insights into support vector machine pattern classifiers using projection-based tour methods. *Proc. KDD 2001*, (2001), 251-256.
7. Dai, J. and Cheng, J. HMMEditor: a visual editing tool for profile hidden Markov models. *BMC Genomics 2008*, 9 (2008).
8. Drummond, C. and Holte, R.C. Cost curves: An improved method for visualizing classifier performance. *Machine Learning*, 65, 1 (2006), 95-130.
9. Evgeniou, T., Pontil, M. and Elisseeff, A. Leave One Out Error, Stability, and Generalization of Voting Combinations of Classifiers. *Machine Learning*, 55 (2004), 71-97.
10. Fails, J.A. and Olsen, D.R.J. Interactive machine learning. *Proc. IUI 2003*, (2003), 39-45.
11. Fei-Fei, L., Fergus, R. and Perona, P. Learning Generative Visual Models from Few Training Examples: An Incremental Bayesian Approach Tested on 101 Object Categories. *CVPRW*, 12 (2004), 178.
12. Fogarty, J., Ko, A.J., Aung, H.H., Golden, E., Tang, K.P. and Hudson, S.E. Examining task engagement in sensor-based statistical models of human interruptibility. *Proc. CHI 2005*, (2005), 331-340.
13. Frank, E. and Hall, M. Visualizing class probability estimators. *Lecture Notes in Artificial Intelligence 2838*, Springer (2003), 168-179.
14. Garner, S.R. WEKA: The Waikato Environment for Knowledge Analysis. *Proc. New Zealand Computer Science Research Students Conference* (1995), 57-64.
15. Grauman, K. and Darrell, T. The Pyramid Match Kernel: Discriminative Classification with Sets of Image Features. *Proc. ICCV 2005*, (2005), 1458-1465.
16. Grimes, D., Tan, D.S., Hudson, S.E., Shenoy, P. and Rao, R.P. Feasibility and pragmatics of classifying working memory load with an electroencephalograph. *Proc. CHI 2008*, (2008), 835-844.

17. Henry, N. and Fekete, J.-D. MatrixExplorer: a Dual-Representation System to Explore Social Networks. *IEEE Trans Visualization and Computer Graphics*, 12, (2006), 677-684.
18. Kapoor, A., Grauman, K., Urtasun, R. and Darrell, T. Active Learning with Gaussian Processes for Object Categorization. *Proc. ICCV 2007*, (2007), 1-8.
19. Kapoor, A. and Horvitz, E. Experience sampling for building predictive user models: a comparative study. *Proc. CHI 2008*, (2008), 657-666.
20. Kim, B., Lee, B. and Seo, J. Visualizing set concordance with permutation matrices and fan diagrams. *Interacting with Computers*, 19, 5-6 (2007), 630-643.
21. Kittler, J., Hatef, M., Duin, R.P. and Matas, J. On Combining Classifiers. *IEEE TPAMI*, 20, 3 (1998), 226-239.
22. Lazebnik, S., Schmid, C. and Ponce, J. Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories. *Proc. CVPR 2006*, (2006), 2169-2178.
23. Mäkinen, E. and Siirtola, H. The Barycenter Heuristic and the Reorderable Matrix. *Informatica* (Slovenia), 29, 3 (2005), 357-364.
24. Meyer, M., Lee, H., Barr, A. and Desbrun, M. Generalized Barycentric Coordinates on Irregular Polygons. *Journal of Graphics Tools*, 7, 1 (2002), 13-22.
25. Moustakis, V. Do People in HCI Use Machine Learning? *HCI*, 2, (1997), 95-98.
26. Oliver, N., Garg, A. and Horvitz, E. Layered representations for learning and inferring office activity from multiple sensory channels. *Computer Vision and Image Understanding*, 96, 2 (2004) 163-180.
27. Patel, K., Fogarty, J., Landay, J.A. and Harrison, B. Examining Difficulties Software Developers Encounter in the Adoption of Statistical Machine Learning. *Proc. AAAI 2008*, (2008), 1563-1566.
28. Picard, R.W. Affective computing. MIT Press, Cambridge, MA, USA, 1997.
29. Rheingans, P. and desJardins, M. Visualizing High-Dimensional Predictive Model Quality. *Proc. VIS 2000*, (2000), 493-496.
30. Saponas, T.S., Tan, D.S., Morris, D. and Balakrishnan, R. Demonstrating the feasibility of using forearm electromyography for muscle-computer interfaces. *Proc. CHI 2008*, (2008), 515-524.
31. Schapire, R.E. The boosting approach to machine learning: An overview. *Nonlinear Estimation and Classification*. Springer (2003).
32. Stiglic, G., Mertik, M., Podgorelec, V. and Kokol, P. Using Visual Interpretation of Small Ensembles in Microarray Analysis. *Proc. CMBS 2006*, (2006), 691-695.
33. Urbanek, S. Exploring Statistical Forests. *Proc. of the 2002 Joint Statistical Meeting*, Springer (2002).
34. Varma, M. and Ray, D. Learning The Discriminative Power-Invariance Trade-Off. *Proc. ICCV 2007*, (2007), 1-8.
35. Wang, J., Yu, B. and Gasser, L. Classification Visualization with Shaded Similarity Matrix, Technical Report, GSLIS University of Illinois at Urbana-Champaign, (2002).
36. Ware, M., Frank, E., Holmes, G., Hall, M. and Witten, I.H. Interactive machine learning: letting users build classifiers. *Int. J. Human-Computer Studies*, 55, 3 (2001), 281-292.
37. Zhang, H., Berg, A. C., Maire, M. and Malik, J. SVM-KNN: Discriminative Nearest Neighbor Classification for Visual Category Recognition. *Proc. CVPR 2006*, (2006), 2126-2136.