Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence

Learning to Learn: Algorithmic Inspirations from Human Problem Solving

Ashish Kapoor, Bongshin Lee, Desney Tan and Eric Horvitz
Microsoft Research
{akapoor, bongshin, desney, horvitz}@microsoft.com

Abstract

We harness the ability of people to perceive and interact with visual patterns in order to enhance the performance of a machine learning method. We show how we can collect evidence about how people optimize the parameters of an ensemble classification system using a tool that provides a visualization of misclassification costs. Then, we use these observations about human attempts to minimize cost in order to extend the performance of a state-of-the-art ensemble classification system. The study highlights opportunities for learning from evidence collected about human problem solving to refine and extend automated learning and inference.

Introduction

The development of machine learning procedures hinges on design choices such as selection of optimization methods, definition of evidential features, and the setting of parameters. Such design decisions rely on expert knowledge and can be viewed as a form of search through a space of alternate methods and models. We have been pursuing opportunities for automating the refinement of machine learning. Our work comes in the context of growing interest in interactive, human-in-the-loop learning, where people iteratively tag, train, and correct classifications in a tight interactive loop. Examples of work in this space include online learning against adversaries, active learning, and contextual bandits.

We examine the prospect of extending machine learning algorithms with methods that learn from people's attempts to optimize the performance of classifiers. Specifically, we collect and use evidence about the steps that human subjects take to minimize misclassification via the use of a tool that provides a visual summary of the performance of a weighted combination of a set of classifiers, as well as controls for modifying the parameters of the classification. We collect observations of subjects' attempts to minimize the costs of misdiagnosis to train classifiers. Then, we use the data autonomously to enhance a machine learning procedure. The methods and results highlight the promise of learning to learn from people, by leveraging human skills at perceiving visual patterns with ease, and codifying evidence about strategies that people formulate to refine the behavior of a system.

We focus on analyzing and learning within EnsembleMatrix (Talbot et al. 2008), a system that allows users to interactively manipulate visualizations of the output of a multiclass classifier system in order to design a more ideal combination of multiclass classifiers. We describe a study we ran to observe human problem solving behavior with the system, review insights we garnered from this study, and present algorithms that we developed with data from human optimization sessions.

Related Work

Recent studies have identified situations in which people produce better classifiers than automatic techniques. Some studies provide users with visualizations about the operation of specific machine learning algorithms along with controls for modifying parameters of the learning procedures and classifications (Ware et al. 2001, Talbot et al. 2008). Related studies have explored the use of people to provide hints to optimize decision trees (Ankerst et al. 1999), naïve-Bayes (Becker et al. 2001), SVMs (Caragea et al. 2001), and HMMs (Dai and Cheng 2008).

Our research is also closely aligned with research on interactive machine learning. Fails and Olsen (Fails and Olsen 2003) assert the potential value of human involvement to provide training data and propose an interactive system that enables users to train, classify, and correct classifications in a real-time iterative loop. In the context of unsupervised learning, interactive clustering using human input has been also proposed (Bilenko, Basu, Mooney 2004, Bekkerman et al. 2007). Beyond label elicitation, related research includes interactive feature discovery for identifying discriminative features

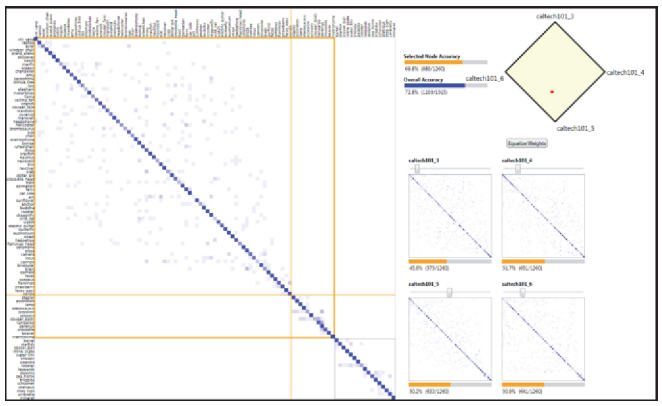


Figure 1: Snapshot of EnsembleMatrix. Leave one fold out confusion matrices of component classifiers are shown in thumbnails on right and their weights in the linear combination are represented by red dot in diamond. Subjects can interact to change weights as well as partition the space of classes. Matrix on left shows confusion matrix of the current ensemble classifier constructed by the user.

(Raghavan, Madani, Jones 2005) and interactive optimization that uses human guidance to maximize human preferences about classification (Kapoor et al. 2010).

The main goal of most of the work mentioned above has been to achieve high classification accuracy rather than to learn from human actions. We note that the problem of reinforcement and imitation learning (Sutton and Barto 1998, Hsiao and Lozano-Perez 2006) are related to our efforts. However, little effort to date has been focused on the task of learning to learn from observations about human efforts to optimize machine classification. In this work, we explore the prospect of developing better classification systems by incorporating insights gleaned by watching people effectively optimize the performance of classifiers in interactive learning scenarios.

Designing Classifiers with EnsembleMatrix

We harness a prototype named EnsembleMatrix that was created earlier for studies of interactive machine learning (Talbot et al. 2008). EnsembleMatrix was designed to assist developers with optimizing the behavior of machine classification in applications. Specifically, the prototype helps developers build an ensemble classifier by combining multiple component classifiers (see Figure 1). 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, the Linear Combination widget on the upper right, and the main Ensemble Classifier view on the left.

EnsembleMatrix represents each classifier using a graphical heat map of its confusion matrix. Confusion matrices allow human observers to use their perceptual capabilities to identify patterns of misclassification. The ordering of any matrix can greatly influence the patterns visible. EnsembleMatrix reorders the main confusion matrix at interactive rates to highlight sets of frequently confused classes (Cuthill and McKee 1969). The prototype provides two basic mechanisms for exploration: (1) a partitioning operation, which divides the class space into multiple partitions, and (2) a linear combination operation that lets the user select weights for each classifier within each partition. Partitioning the class space separates the data instances into two subsets, allowing the user to develop predictors specialized for each subset. The two subset classifiers are restricted to only predicting classes within their partition. The second interaction mechanism allows the user to quickly manipulate a subspace of the linear combination of component classifiers using a simple two-dimensional interpolation control. Partitioning and linear combination operations can be done recursively an arbitrary number of times and in any order, leading to a large number of possible refinement strategies.

Talbot et al. (Talbot et al. 2008) observed that when asked to find configurations of partitions and weight parameters users could find reasonable solutions fairly quickly. We note that the task of finding good parameters is non-trivial due to the size of the search space. However, the efficiency with which humans perform this task raises two questions. First, what are the key properties of the strategies employed by humans? Second, can we train machine learning systems to learn aspects of human strategies in order to build better predictors? We explore these questions in depth in the rest of the paper.

Studies of Human Optimization Strategies

In the first stage of this research, we conducted an experiment to explore the different strategies that people use with EnsembleMatrix to build models.

Datasets and Individual Classifiers: To cover a wide range of problems in terms of the different types of datasets, we considered three different multiclass classification problems. We studied people working with (1) Newsgroup (3000 examples, 20 classes), (2) Multipie (Gross et al. 2008) (3000 examples, 150 classes), and (3) Caltech-101 (Fei-Fei, Fergus, Perona 2006) (3030 examples, 101 classes) datasets. The goal for the Newsgroup dataset is to categorize documents into 20 different topic classes. The task for the Multipie collection of face images is to recognize the identities of the faces. The task for Caltech-101 challenge is to classify images into 101 categories. For each classification task, we consider four base-level classifiers that were trained using different sets of features. In particular, we generate four different kernels to induce four classifiers. For the Newsgroup dataset, we consider three polynomial kernels with degrees 1, 2, 3 and an RBF kernel. Similarly, for the Multipie data, we first perform PCA to project the images onto 300 dimensions, and then induce four different RBF kernels using four successive partitions of the PCA representation. For the Caltech-101 data, we consider four different kernel representations based on gray level intensities, color intensities, and horizontal and vertical edge orientations.

	Before Interaction	After Interaction	Improvement
Training Data	78.1	80.4	2.3
Test Data	78.2	79.7	1.5

	Before Interaction	After Interaction	Improvement
Training Data	77.4	78.6	1.2
Test Data	79.5	79.0	-0.5

Figure 2: Instance from study that illustrates how good (left) and poor (right) interaction decisions can help and hurt in classifying test data. Gains on training accuracy over Flat Average models are obtained for both cases. Gains in accuracy for the test set are observed in the case where the subject has also tried to minimize deviations from the diagonal, in addition to maximizing the on diagonal elements.

For each of the datasets, we perform a 50-50 train and test split. We then use Gaussian Process regression models (Rasmussen and Williams 2006) to compute 10-fold cross-validated mean predictions over the training set. These cross-validated predictions over the training set are used to generate the confusion matrices that are used in the EnsembleMatrix studies. We note that cross validation ensures that the resulting model will lead to good generalization performance on the test set.

Participants and Apparatus: We recruited 10 participants from an industrial research institution. Participants spanned a wide spectrum of expertise, ranging from machine-learning researchers to ones that had little exposure to machine learning. We provided participants with basic instructions on how to use the EnsembleMatrix system, mainly on partitioning and controlling linear combinations within each partition. We told the participants that their task was to develop a new, more accurate, ensemble classifier by exploring and combining four different component classifiers. Each participant worked on the three datasets which were presented in the same order: the Caltech-101, Multipie, and then Newsgroup dataset. For each dataset, we asked participants to try the same task twice. At the end of each trial, we informed the participants of the result accuracy with the test dataset so that they could adjust their strategy in the following trials. We believed that participants could learn a better strategy not only from the previous dataset but also from the first trial. Each task ended when the participant assessed that they could no longer improve the ensemble classifier. Each session (which included six EnsembleMatrix tasks) lasted approximately one hour.

Results and Observations: Table 1 shows the best recognition accuracy (out of the two trials) achieved by each participant on the different tasks. We observed that most of the participants perform this task fairly well regardless of their familiarity with machine learning. We also note that resulting models were fairly different from one another, resulting in a spectrum of accuracies. We were interested in seeing if the models designed by people could achieve better classification than a standard machine-learning baseline. In particular, we compared the performance of the models built by people with a non-hierarchical classifier (denoted as Flat Average) that first computes equally weighted linear combination of the component classifiers and then performs one-vs-all classification. The underlined entries in Table 1 depict instances when a human-designed classifier was better than a classifier built autonomously using the Flat Average.

We observed that several participants could generate classifiers with performance boosts over the baseline for the Caltech-101 (4 participants) and the Newsgroup data (5 participants). Only one participant achieved better performance than the Flat Average on the Multipie data,

Table 1: Best test accuracy obtained by the ten participants on the three datasets. The bold underlined accuracies indicate boost over the baseline Flat Average accuracy (mean baselines: 79.39, 79.63, 91.27 for Caltech 101, Newsgroup, and Multipie respectively).

Subject	Caltech-101	Newsgroup	Multipie
1	79.14	**79.73**	90.4
2	79.47	**78.67**	**91.6**
3	76.77	79.6	91.27
4	79.08	**80.6**	90.73
5	79.74	79.53	90.07
6	**81.78**	79.4	90.8
7	78.88	79.8	89.93
8	**81.3**	78.87	91.13
9	**80.46**	**80.33**	91.27
10	**79.93**	**80.13**	90.87

although the rest of the participants created classifiers with a relatively high accuracy. We pursued insights about why some participants were able to build better models than others. Figure 2 illustrates instances of a good and poor model trained by a human. Although both models show gains over the Flat Average baseline, per their accuracy on the training set, the models behave differently on the test set. We noted several key differences between models that provide good classifications versus the poor ones. One key difference is the way that the partitions are formed. For instance, in the classifier portrayed in Figure 2, the partitions are of equal size in the poor model whereas the partitions in the good models are imbalanced. The imbalanced partitions are cleaner (showing less confusion among partitions) in the good model. It is straightforward to prove that if a partition is clean, then the additional steps taken by the user can only increase the accuracy.

We also observed that, while interacting with EnsembleMatrix, study participants continued to make trades between structure and accuracy. Sometimes participants might choose weights that lead to lower overall accuracy but that would show better structure in the confusion matrix, and thus lead to cleaner partitions. When poor models were constructed, we found that participants often chose to optimize for maximum accuracy. While such steps might be locally optimal, overall they could lead to poorer models in the end.

Finally, we noted that, instead of continuing to partition the space until the very end of a session, participants stopped partitioning at some point in the refinement of the matrix. Such stopping, based on the state of confusion matrix, can have a regularizing effect and prevent models from overfitting on training data. As shown as an example of a good classifier in Figure 2, adept participants appear to be good at judging when to stop. Although there are partitions of many different sizes, the participant chooses to stop at reasonable points leading to good generalization.

Our observations indicate that participants who were inclined to build good models attempt to balance the constraints of (1) finding clean partitions, (2) choosing weights that reveal structure, and (3) determining when to stop. And at every stage in EnsembleMatrix, people face decisions in a space of an exponential number of options. We next describe a methodology where we learn how to autonomously mimic choices that people who are successful with use of EnsembleMatrix make, with the goal of developing better hierarchical ensemble classifiers.

Learning by Observing Good Behaviors

We now describe a methodology to mine the data collected in the user study described below. Three key decisions are made at each step in the refinement of EnsembleMatrix: (1) choice of appropriate partition, (2) choice of weights for component classifiers, and (3) decision about halting the partitioning. These steps are interdependent, and iterative refinements of the misclassification matrix based on sequences of transitions among these three actions define a massive space of possibilities. We sought to encode and embed human expertise to help an automated system cut through this large search space of interdependent partitioning, weighting, and stopping.

To acquire the human expertise at guiding the search through the large parameter space, we build two decision-making modules that determine (1) quality of a partition (denoted as $Quality(.)$) and (2) whether the system should stop partitioning (denoted as $Stop(.)$). Theoretically, we can enumerate all possible combinations of weights and partitions of the resulting confusion matrix and apply our learned decision modules in order to further split the problem or to halt. However, as such exhaustive enumeration is infeasible, we formulate a tractable approximation. First, since the number of component classifiers is fairly small (four), we can sample the whole space of weight parameter[1]. Second, instead of considering all possible partitions of a given set of weights, we use an existing spectral clustering algorithm (Zelnik-Manor and Perona 2004) to produce partitions.

Algorithm 1 shows the pseudo-code of the procedure. We use the methodology described above to generate a proposal of partitions for each set of weights. These proposals in turn can be ranked by the trained quality function ($Quality(.)$). The highest ranked proposal is selected by the method and the weights are further tuned using an optimization criterion. Also, for the best choice of proposal, we further evaluate if the trained stopping criterion ($Stop(.)$) returns true. The procedure can either repeat recursively or end depending on the result of the stopping function. We next describe how we learn the $Quality(.)$ and $Stop(.)$ criterion.

[1] In our work, we sample between [0..1] with the step size of 0.1.

Learning to Judge Good Partitions: The $Quality(.)$ function is learned as a one-class classifier from the data collected during the user study. In particular, we look only at the partitions generated by the participants that performed better than the baseline models. We extracted an eight-dimensional feature vector from each of these partitions which characterized structural properties of the confusion matrix under such partitioning. The features consisted of the proportion of data points that were grouped correctly within their corresponding split (one feature for each of the two diagonal regions) and data points that were grouped incorrectly (one feature for each of the two off-diagonal regions). The other four features encoded accuracies within each group and the sizes of each of the partition.

As we only had access to examples of partitions provided by participants, we trained a one-class classifier using a linear Bayes Point Machine (BPM) with probit noise model (Minka 2001) trained with expectation propagation. The BPM integrates over the parameter space of the classifier which in turn makes it capable of handling a one-class problem. Given a test vector, the BPM returns the probability that the corresponding proposal of partition is a good choice. This probability can be used to rank all proposals being considered. Note that instead of learning the quality function, we can use other heuristics that measure various notions of the quality of clusters. For example, the self-tuning spectral clustering (Zelnik-Manor and Perona 2004) provides a quality measure based on spectral properties of the partition. We performed comparisons against such heuristic (see Section 5) but found learning from humans with the eight features we have described to be superior. It is also possible to encode these algorithm-specific heuristics as features in the proposed framework.

Learning When to Stop: The $Stop(.)$ function is trained as a binary classifier via fusing the data observed in the user study. We consider only the data from the participants for cases that showed boosts over the baseline classifier. We generate training sets by considering all the partitions and then featurizing them as described earlier. We trained a linear BPM (a linear Gaussian Process classifier) by considering all vectors except the terminal partitions labeled as -1. Given a test vector, the $Stop(.)$ function provides the probability that the corresponding partition is a terminal one.

Optimizing Weights for Separation of Clusters: Once the partition is determined, we need to find weights that maximize separation between these clusters. To this end, we use the softmax function to approximate probabilities that a data point is assigned to a cluster. We denote the vector comprising of outputs of all component classifier for a class c as $o_i^{(c)}$. Then, we define probability that the i^{th} data point is classified into the correct cluster as follows:

$$p_i = \frac{\sum_{c \in cluster_i} e^{w^T o_i^{(c)}}}{\sum_{c \in all\ classes} e^{w^T o_i^{(c)}}}$$

We invoke the principal of maximum likelihood to find the optimal weights. In particular, we minimize the following objective to tune the weights:

$$w^* = argmin_w \sum_{i \in Data} -\log p_i \qquad (1)$$

Algorithm 1 Learning Multilevel Ensemble Model

```
function ModelOut = LearnTree(Classes, Data)
ModelOut.wt  {}, ModelOut.left  {}, ModelOut.right  {}
if (length(Classes) > 1)
    for all weight settings
        Compute Confusion Matrix C
        Partition  SpectralPartition(C)
        Q_partition  Quality(Partition)
    end for
    [Classes_left, Classes_right]  argmax Q_partition
    If Stop(Partition)  false
        ModelOut.wt
            argmax_wt Objective(Classes_left, Classes_right, Data)
        ModelOut.left  LearnTree(Classes_left, Data)
        ModelOut.right  LearnTree(Classes_right, Data)
    end if
end if
Return ModelOut
```

Experiments and Results

We carried out experiments to study if good clustering and stopping criteria can be learned from human actions and if the resulting classifier would be able to perform well on the test data. We also explore how the human-learned model compares with reasonable ML attempt; that is, we compare the learned scheme (denoted as *Human Learned*) with four methods that rely on existing techniques:

1) *Non hierarchical Average Classifier (Flat Average):* Sum equally weighted component classifiers to do one-vs.-all classification. This is the simplest case and serves a baseline.
2) *Non hierarchical Trained Classifier (Flat Trained):* Sum of weighted component classifiers to do 1-vs-all classification. The weights are trained by maximizing the objective in Equation (1) assuming each class is a separate cluster.
3) *Hierarchical (Hierarchy Full):* Same method as described in Algorithm 1 with a learned clustering quality criterion (Zelnik-Manor and Perona 2004) and without stopping the splitting until the leaf nodes are completely separable.
4) *Hierarchical with Stopping (Hierarchy Stop):* Same as Algorithm 3 with a stopping heuristic that stops partitioning if the learned quality criterion is less than a threshold (0.99).

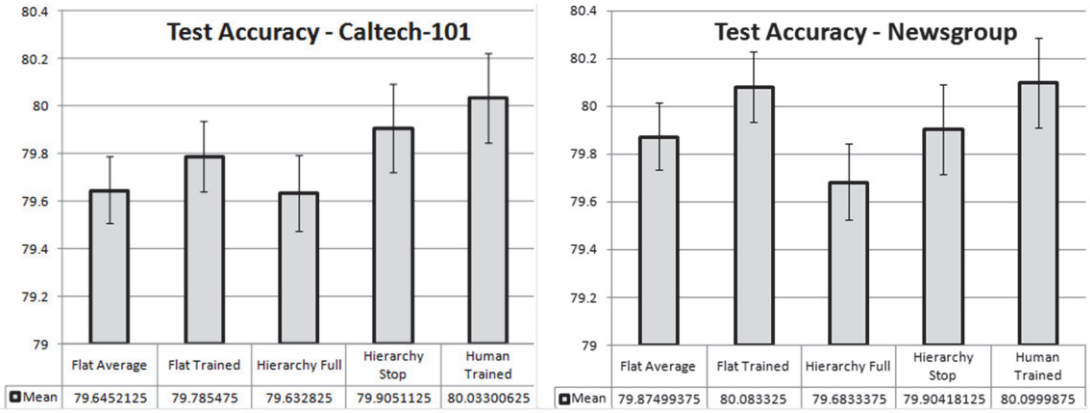


Figure 3: Average classification accuracy over 16 train tests splits using different methods. Error bars show standard error.

First, we explore if the proposed scheme performs better on the test examples. We note that all of the human interaction as well as all of the training steps are limited to the train-split of the data and neither the human nor the training procedure has seen these test data points. Further, in order to train the model for determining cluster quality and the stopping criterion, we only use data from the individuals that had showed boost over the baseline.

Figure 3 shows the recognition performance of the proposed human-learned scheme as well as the four alternatives on the Caltech-101 and Newsgroup datasets. We did not perform this test for the Multipie dataset as we did not have data from humans that showed gain (we tackle Multipie data with transfer learning later in this section). We plot the mean accuracy and standard error over 16 different train and test splits; the human-trained scheme consistently outperforms the Flat Average baseline on all the datasets and shows considerable gains over the others. Furthermore, we observe that Hierarchy Full consistently results in worse performance indicating that knowing when to stop is important. This hypothesis is further confirmed by observing that Hierarchy Stop fairs much better. These results suggest that one of the reasons humans do well in the task is that they apply expertise about when to stop, perhaps with the aid of recognizable visual patterns, versus the poor competencies with analogous "sensing" of state employed within the learning procedures.

Next, we examine the patterns of partitioning that the different methods generated. Figure 4 indicates different instances of resulting training confusion matrices with inferred partitioning of the Caltech-101 data by the three hierarchical methods (Human Learned, Hierarchy Full, and Hierarchy Stop). We note that the partitioning by Hierarchy Full suffers from over partitioning of the space, resulting in poor test accuracy (79.41%). We can see that the human-learned partitioning is similar to the good partitioning depicted in Figure 2. Although the Hierarchy Stop works reasonable, it still appears to suffer from over fitting due to over partitioning. The observed test accuracy for this instance was 80.01% and less than the one achieved by Human Learned (80.67%).

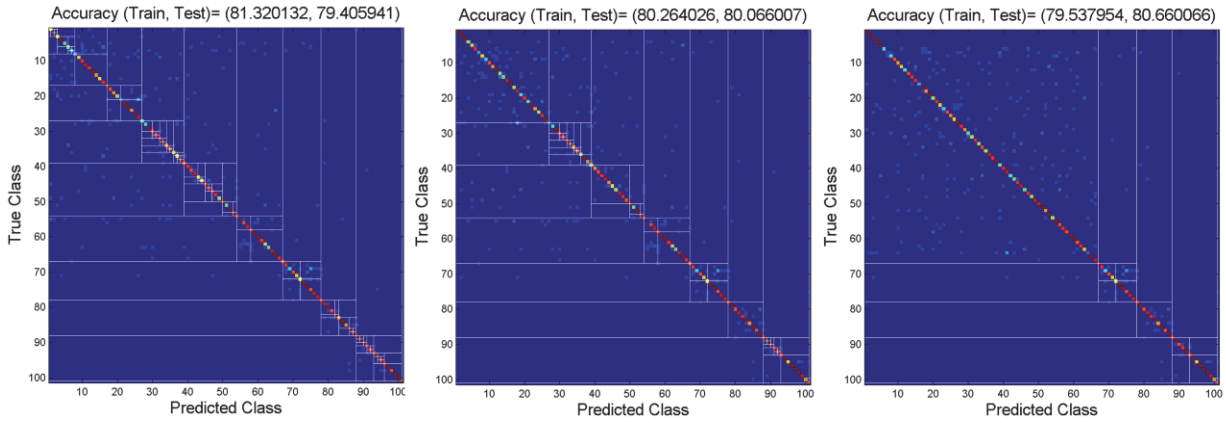
Table 2: Boost over baseline (Flat Average) when Human Learned is trained on one dataset and applied to another.

Train / Test	**Caltech**	**Newsgroup**	**Multipie**
Caltech	0.39	0.27	NA
Newsgroup	0.23	0.23	NA
Multipie	0.27	0.15	NA

Finally, we tackle the Multipie data, where we found relatively poor performance by participants, perhaps due to high baseline recognition accuracy. We address the Multipie challenge in the context of explorations about the transferability of learning from humans about the partition and stopping criterion across datasets. In particular, we train our human-learned model using one dataset but then use it on unseen test data. Table 2 depicts results where we show average boost in accuracy (from the Flat Average baseline) when the human-learned model is trained on different datasets. The diagonal is the original case where the two sets match. Surprisingly, we still observe a boost in the off-diagonal entries, including for the Multipie case. Such a boost suggests that common features may exist across different learning algorithms that can be learned by observing people while they perform classification optimization for different tasks. We believe that these results may have far reaching implications in that they highlight potentially task-independent capabilities derived from people that can be harnessed by machine learning algorithms to effectively cut through large parameter spaces.

Summary and Future Work

We pursued opportunities with extending machine learning procedures by collecting data and learning about the strategies of people seeking to optimize the performance of classifiers. In particular, we observed the activities of subjects interacting with a visual representation of the performance of ensemble classifiers. Data was collected about how people define multi-level hierarchies in a multiclass classification problem. Our findings suggest that people frequently can find good solutions without doing



(a) Hierarchy Full (c) Hierarchy Stop (a) Human Learned

Figure 4: Training confusion matrices and resulting partitioning for the Caltech 101 dataset, obtained by applying three hierarchical methods. Test accuracy obtained by the human learned method is the best of the three methods. We find that the result for the Human Learned case is most similar to results obtained when people make good decisions (Figure 2).

exhaustive search. We found evidence that participants' success at this task is linked to the skills they have with judging the quality of partitions and with determining when to halt partitioning. We presented and evaluated a methodology for learning such human competencies by collecting data about human optimization activities and then using a machine learning algorithm to construct classifiers that are used at run time as components of a new learning procedure. An evaluation demonstrates that we can enhance machine learning with insights gleaned from human problem solving. We also showed the potential to share the learned competencies across classification tasks.

Our work is an initial attempt to design a pipeline that enables machine-learning systems to learn from humans. We aim to identify other forms of interaction, as well as rich features for capturing behaviors, with the goal of embedding additional human insights into machine-learning procedures.

References

20 Newsgroups. http://kdd.ics.uci.edu/databases/20newsgroups/20newsgroups.html.

Ankerst M., Elsen C., Ester M. and Kriegel H. P. (1999). Visual Classification: An Interactive Approach to Decision Tree Construction. KDD.

Becker B., Kohavi R. and Sommerfield D. (2001). Visualizing the Simple Bayesian Classifier. Information Visualization in Data Mining and Knowledge Discovery. Eds. Fayyad et al.

Bekkerman R., Raghavan H., Allan J. and Eguchi K. (2007). Interactive Clustering of Text Collections According to a User Specified Criterion. IJCAI.

Bilenko M., Basu S. and Mooney R. J. (2004). Integrating Constraints and Metric Learning in Semi Supervised Clustering. ICML

Caragea D., Cook D. and Honavar V.G. (2001). Gaining Insights into Support Vector Machine Pattern Classifiers using Projection Based Tour Methods. KDD.

Cuthill E., and McKee J. (1969). Reducing the Band width of Sparse Symmetric Matrices. ACM National Conference.

Dai J. and Cheng J. (2008). HMM Editor: a visual editing tool for profile hidden Markov models. BMC Genomics 9.

Fails J. A. and Olsen D.R.J. (2003). Interactive machine learning. IUI.

Fei Fei L., Fergus R. and Perona P. (2006). One Shot Learning of Object Categories. PAMI 28(4).

Hsiao K. and Lozano Perez T. (2006). Imitation Learning of Whole Body Grasps. IROS.

Gross R., Matthews I., Cohn J. F., Kanade T. and Baker S. (2008). Multi PIE. Int. Conference on Automatic Face and Gesture Recognition.

Kapoor A., Lee B., Tan D. and Horvitz E. (2010). Interactive Optimization for Steering Machine Classification. CHI.

Meyer M., Lee H., Barr A. and Desbrun M. (2002). Generalized Barycentric Coordinates on Irregular Polygons. Journal of Graphics Tools 7(1).

Minka T. P. (2001). Expectation Propagation for Approximate Bayesian Inference. UAI.

Raghavan H., Madani O. and Jones R. (2005). InterActive Feature Selection. IJCAI.

Rasmussen C. E. and Williams C. K. I. (2006). Gaussian Processes for Machine Learning. The MIT Press.

Sutton R. S. and Barto A. G. (1998). Reinforcement Learning: An Introduction. The MIT Press.

Talbot J., Lee B., Kapoor A. and Tan D. (2008) EnsembleMatrix: Interactive Visualization to Support Machine Learning with Multiple Classifiers. CHI.

Ware M., Frank E., Holmes G. Hall, M. and Witten I. (2001). Interactive machine learning: letting users build classifiers. IJHCS 56(3).

Zelnik Manor L. and Perona P. (2004). Self Tuning Spectral Clustering. NIPS.