

Active Learning in the Presence of Unlabelable Examples

Dominic Mazzoni¹ and Kiri Wagstaff¹

Jet Propulsion Laboratory, California Institute of Technology,
Pasadena, CA 91109, USA,
{Dominic.Mazzoni,Kiri.Wagstaff}@jpl.nasa.gov

Abstract. We propose a new active learning framework where the expert labeler is allowed to decline to label any example. This may be necessary because the true label is unknown or because the example belongs to a class that is not part of the real training problem. We show that within this framework, popular active learning algorithms (such as **Simple**) may perform worse than random selection because they make so many queries to the unlabelable class. We present a method by which any active learning algorithm can be modified to avoid unlabelable examples by training a second classifier to distinguish between the labelable and unlabelable classes. We also demonstrate the effectiveness of the method on two benchmark data sets and a real-world problem.

1 Introduction

For many classification problems, class labels must be provided by a domain expert and are therefore expensive to acquire. The standard approach to classification assumes that all labeled examples are provided up front. Active learning [1] attempts to reduce this burden by incrementally selecting only the most useful items for labeling. For this to be effective, the expert labeler is *required* to assign a valid label to whatever items the active learner selects. However, there are cases where an item does not fall into any of the specified classes, or the expert may be unsure about its classification. Requiring the expert to assign labels for these items introduces unnecessary errors into the learning process. Even if the expert is allowed to abstain from assigning a label, we find that this situation can cause typical active learning algorithms to perform worse than random selection.

As a concrete example, consider the problem of training an automatic cloud mask from satellite images. Given image data from NASA’s Multi-angle Imaging SpectroRadiometer (MISR) instrument [2], the goal is to correctly classify each pixel as either “clear” or “cloudy”. Figure 1(a) shows a scene captured by MISR over the Sahara desert, containing areas of clouds, clear desert, and a third region contaminated by dust. The expert labeler can easily identify the pixels in region 3 as being neither clear land nor clouds, and therefore irrelevant to the classifier goal. Assigning them to either class would be misleading. Figure 1(b) shows the accuracy of three algorithms when evaluated on the problem of separating cloudy and clear pixels. These algorithms will be described more fully in Section 2.1. The

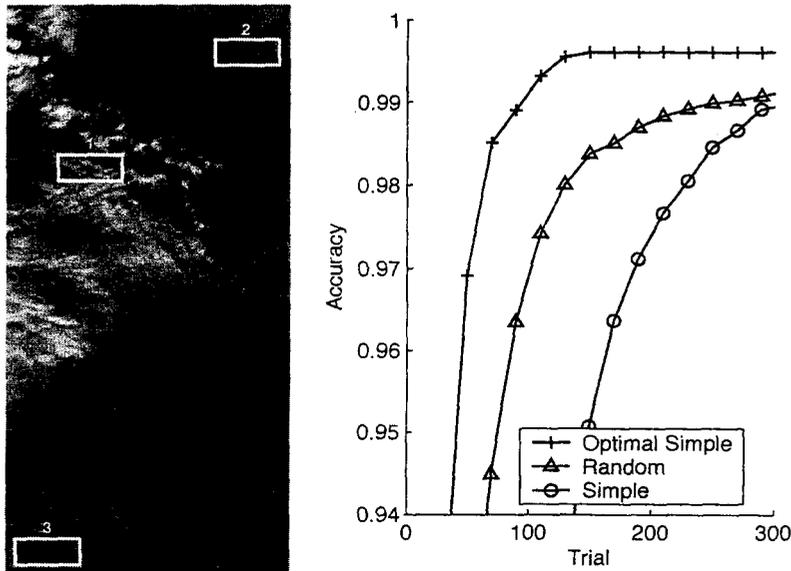


Fig. 1. Part of the Sahara Desert captured by MISR’s 70-degree forward-looking camera on February 6, 2004. The three boxes represent regions that are (1) Cloudy, (2) Clear, and (3) Other. On the right is the accuracy achieved by Simple when trained on regions 1 and 2 (“Optimal”) as well as Random and Simple when trained on all 3 regions (region 3 marked as unlabelable). All results were averaged over 100 trials.

first curve is “Optimal Simple”, which is the performance achieved by the Simple Margin active learning algorithm [3] when trained only on cloudy and clear pixels. “Simple” shows the performance of the same algorithm when unlabelable (dusty) pixels are included in the training set, and “Random” indicates random selection in the presence of unlabelable pixels. The three algorithms were tested only on labeled pixels. Simple’s learning rate slowed down significantly because it devoted a large percentage (79%) of its queries to unlabelable pixels. In fact, it performed much worse than random selection. We will discuss this phenomenon in more detail in Section 2.3.

This paper seeks to address the limitations of current active learning methods when unlabelable items are present via two major contributions. First, we propose an active learning framework where “unknown” is a valid response from the trainer. Second, we present active learning methods that can make effective use of this information to improve the efficiency and reduce the cost of active learning. Section 2 formalizes the problem of performing active learning and highlights the modifications that are necessary when some labels are unknown. We present our solution to this problem in Section 3, experimental results in Section 4, and key conclusions in Section 5.

2 Active Learning

We focus on *pool-based* active learning, where the learner has access to a (fixed) pool of items for which it can request labels. We assume the existence of a pool $\mathcal{U} = \{x_i\}$ of unlabeled items. Each x_i is a d -dimensional vector in Euclidean space, and the items are assumed to be i.i.d. according to an unknown fixed distribution $P(x)$. In addition, there exists a binary classification label $y_i \in \{\pm 1\}$ for each x_i that is available, upon request, from the expert labeler. The expert can be a human or an automated labeler that incurs some cost per query; we refer to the expert's labeling of x as $f(x)$. Let \mathcal{L} be the set of items for which the learner has already requested labels. On each trial, the active learner selects an unlabeled item x from \mathcal{U} and receives its label, $y = f(x)$. The learner then applies its classifier learning algorithm to $\mathcal{L} \cup \{(x, y)\}$ to train a new model.

In this section, we first describe a variety of algorithms that have been proposed and then discuss how the active learning problem changes when unlabeled items may exist.

2.1 Active Learning Algorithms

Although active learning is not restricted to any single inductive learning technique, much of the recent work in this area has focused on active learning for support vector machines (SVMs) [4] due to their strong performance on a variety of problems. An SVM is a binary classifier that constructs a hyperplane in d dimensions to separate the two classes. In particular, it seeks the hyperplane that will maximize the *margin*, or distance between each class and the hyperplane. For classes that are not linearly separable, the SVM implicitly maps each point into a higher-dimensional space via a *kernel function*, which often improves separability. All active learning methods seek to select the item x which, when labeled, will provide the most benefit in terms of an increase in accuracy achieved by the model constructed from $\mathcal{L} \cup x$.

1. **Random**: This method selects $x \in \mathcal{U}$ randomly. It is equivalent to “passive” learning, as the algorithm has no input on the order of the items it sees. It serves as a baseline for comparison with active learning algorithms.
2. **Simple Margin (“Simple”)** [3]: **Simple** assumes that the item closest to the current hyperplane is likely to be the most helpful for updating the hyperplane in the next trial. This algorithm ranks each example by its distance from the hyperplane (computed using the absolute value of the SVM output on this example) and then chooses the minimum.
3. **MaxMin Margin** [3]: Rather than guessing that the item closest to the hyperplane will yield the most information, this method aims to empirically test which item will be most effective. Let m be the size of the separation between the positive and negative classes (the margin) for a given SVM. Then, for each $x \in \mathcal{U}$, this method trains two SVMs: one on $\mathcal{L} \cup \{(x, +1)\}$ (yielding m^+) and one on $\mathcal{L} \cup \{(x, -1)\}$ (yielding m^-). Finally, **MaxMin** selects x such that $\min(m^-, m^+)$ is maximized. For efficiency, we only considered 20% of the queries each trial.

4. **Diverse** [5]: This algorithm attempts to avoid choosing too many similar queries by increasing the diversity of the chosen examples. The diversity of $\mathcal{L} \cup x$ can be maximized by selecting x whose maximum normalized kernel distance to all of the other labeled examples is minimized. The complete method uses a weighted sum of diversity and hyperplane distance, controlled by a parameter λ , where $\lambda = 0$ is the equivalent of focusing solely on diversity and $\lambda = 1$ is the same as **Simple**. We determined that $\lambda = 0.5$ worked well for our experiments.
5. **Kernel-farthest-first (KFF)** [6]: This method maximizes the minimum distance from x to all $x' \in \mathcal{L}$. This algorithm only works well for a few special XOR-type problems, but we included it because it gave interesting results in one of our experiments (Section 4).

2.2 Probabilistic Active Learning

Because each of the algorithms described above is a heuristic, it may not always be advantageous to select the top-ranked query. We considered a variant that assigns a probability to each query based on the ranking produced by the algorithm and then randomly chooses an item according to this distribution. Specifically, we normalized the results of each algorithm **ACTIVE** to a value between 0.0 and 1.0 (to be maximized). We then assigned the probability of picking each query using a Gaussian function, chosen so that queries with a value of 1.0 are about 100 times more likely to be picked than ones with a value of 0.0:

$$P(q_i) = e^{-4.6(\text{ACTIVE}(x)-1)^2}$$

The next query is chosen according to this distribution. We will use the abbreviation **Prob.** before the name of an active learning algorithm to designate the probabilistic variant.

2.3 Active Learning with Unlabelable Items

Each of the existing active learning methods uses a different heuristic to select the next item for labeling, but they were all designed with the assumption that the label exists. As shown in Figure 1, when this assumption fails, active learning can perform worse than random selection. Therefore, we propose a new model of the active learning problem that allows for the possibility of unlabelable items.

We model the expert labeler as a function f' that maps items to three possible values, $y_i \in \{-1, 0, +1\}$. A value of 0 indicates that the label is unknown. Again, on each trial, the active learner applies a selection function to choose an unlabeled item x from \mathcal{U} . The learner proceeds normally unless $f'(x) = 0$, in which case it acquires no new information and must wait until the following trial to make a new request. There are several possible ways to handle this situation.

First, the learner could ignore the unlabelable examples. As previously discussed, this is undesirable because it can cause active learning to perform worse

than random selection in terms of learning speed. For example, the **Simple** algorithm deliberately selects items that are ambiguous (close to the current SVM hyperplane). When the ambiguous items are not members of the two classes the SVM is attempting to learn, **Simple** will query exactly the items that help it the least. This is the case for the MISR example shown in Figure 1.

Second, the learner could assign all unlabelable examples to one of the valid classes, such as the positive class. This introduces noise into the labels. If the unlabeled examples overlap both classes, the SVM is forced to find a much more complex solution to successfully separate the classes. Even if they do not overlap, the positive class is likely to have worse generalization performance, since it includes items that are not true positives.

Third, another strategy would be to train a multi-class classifier, where the unlabelable examples would be assigned to a separate class. For some problems this could be effective, but it is not optimal as it is likely to cause the learner to spend time querying and modeling the third class.

Instead, we propose a model where the learner collects unlabeled items x in a set \mathcal{R} of rejected queries, which it can potentially use to try to avoid querying unlabelable examples in the future. Because the cost of acquiring y must be paid regardless of its information value, it is to the learner’s advantage to avoid requesting items that are unlabelable. Correctly predicting which items are likely to be labelable is an independent learning problem for which good performance will enable much more efficient active learning.

3 Solution: Biasing Toward Labelable Examples

We propose an active learner that trains an additional classifier to distinguish between the set of examples in \mathcal{L} (the labeled set, both positive and negative examples) and the examples in \mathcal{R} (the set of queries that were rejected as unlabelable). The learner then uses this extra classifier, C , to influence its decision about which example should be chosen next. We chose to make the positive class be the labelable class (trained from the examples in \mathcal{L}) and the negative class be the unlabelable class (trained using \mathcal{R}).

This method can be used to augment any existing active learning algorithm, rather than relying on any one particular algorithm. We formulated two approaches to using the results of C improve ACTIVE’s performance:

1. **Reject Unlabelable (RU)**: Evaluate all of the examples in the pool using the new classifier C . Create a new pool $\mathcal{P}' \subseteq \mathcal{P}$ of the examples which C considers to be labelable, and run ACTIVE on \mathcal{P}' to choose the next query.
2. **Labelable Bias (LB)**: Instead of using a hard cutoff that completely eliminates some examples that may actually be labelable, we can use C as a probabilistic weighting that affects the relative score of each example in the pool. **LB** first uses ACTIVE to rank all of the examples in the pool, then normalizes the scores so that they are all in the range $[0, 1]$. Let $\text{norm}[\text{ACTIVE}(x)]$ be the normalized result of ACTIVE on example x . We then maximize the product

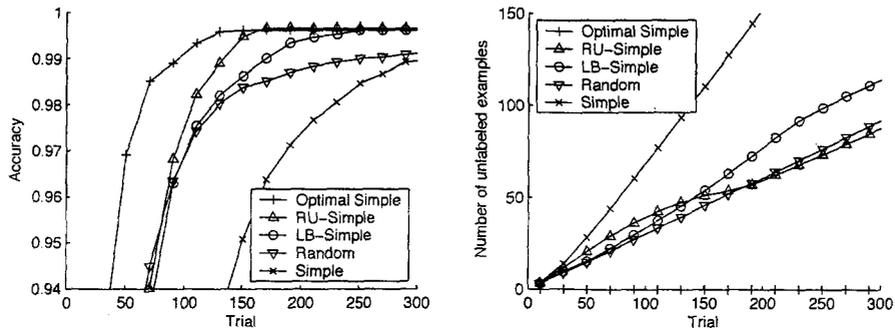


Fig. 2. MISR results for two methods that attempt to avoid unlabeled examples. “Optimal Simple” is included for comparison. All results are averaged over 100 trials.

of $\text{norm}[\text{ACTIVE}(x)]$ and $P(\text{labelable}|x)$. When C is an SVM, $P(\text{labelable}|x)$ can be derived by taking the SVM output, clipping values outside the range $[-1, 1]$ and mapping to the range $[0, 1]$, or formally:

$$LB[\text{ACTIVE}](x) \equiv \text{norm}[\text{ACTIVE}(x)] \cdot \begin{cases} 1 & \text{if } C(x) \geq 1 \\ \frac{1}{2}(1 + C(x)) & \text{if } -1 < C(x) < 1 \\ 0 & \text{if } C(x) \leq -1 \end{cases}$$

In our experiments, we trained an SVM for C using the same training parameters (e.g. kernel) as was used for ACTIVE.

Figure 2 shows the results obtained by both of these approaches, when ACTIVE is **Simple**, on the MISR cloud classification data set shown in Figure 1. The figure also includes the results of **Simple**, **Random**, and **Optimal Simple** for comparison. The figure shows both the accuracy and the average number of unlabeled examples chosen at each trial. Intuitively, the more unlabeled examples that are chosen by the learner, the worse its overall accuracy should be. For example, **Simple** selected the largest number of unlabeled examples, and it had the worst performance. Figure 2 shows that both **RU-Simple** and **LB-Simple** significantly reduced the number of unlabeled examples chosen, and they both had higher performance than **Random** and **Simple**.

4 Experimental Results

When the expert labeler has the option to abstain from providing a label to an active learner, it is important for the learner to be able to avoid making queries of that kind so as to maximize the amount of information it receives. To evaluate the effectiveness of our proposed approach, we conducted experiments on several data sets, with and without unlabelable items.

4.1 Evaluation Metrics

To compare the effectiveness of different active learning algorithms, we use three evaluation methods. *Deficiency* [6] seeks to quantify the learner’s overall improvement in learning efficiency as compared to passive learning. Let $\text{Acc}_t(\text{ACTIVE})$ be the level of accuracy achieved by active learner **ACTIVE** at trial t . After n trials ($1 \leq n \leq |\mathcal{U}|$), we calculate the deficiency of **ACTIVE**, normalized against the performance of **Random**, as

$$\text{Def}_n(\text{ACTIVE}) \equiv \frac{\sum_{t=1}^n \text{Acc}_n(\mathbf{Random}) - \text{Acc}_t(\text{ACTIVE})}{\sum_{t=1}^n \text{Acc}_n(\mathbf{Random}) - \text{Acc}_t(\mathbf{Random})}. \quad (1)$$

Deficiency compares the area under the learning curve to the accuracy achieved by **Random** after n trials. Smaller values of deficiency (which can be negative when $n < |\mathcal{U}|$) indicate more efficient active learning. The deficiency of **Random** is, by definition, always 1.0.

In some cases, however, we are more interested in how quickly the learner reaches a specific level of accuracy (ρ). We define ρ -trials(**ALG**) to be the number of trials required by **ALG** to reach accuracy ρ and ρ -speed to be the factor of improvement with respect to **Random**, where larger values of ρ -speed indicate faster convergence:

$$\rho\text{-speed}(\text{ACTIVE}) \equiv \frac{\rho\text{-trials}(\mathbf{Random})}{\rho\text{-trials}(\text{ACTIVE})} \quad (2)$$

The preceding algorithms are useful for comparing the relative efficiency of different active learners. However, we also require a measure that indicates how effectively an active learner performs in the presence of unlabeled items. We define $\text{OptAcc}_i(\text{ACTIVE})$ as the accuracy that would be obtained by **ACTIVE** after i selection trials, if the algorithm could perfectly avoid unlabeled items. Then the *label efficiency* (Leff) is defined as the algorithm’s performance relative to OptAcc , with **Random** as a lower bound:

$$\text{Leff}_n(\text{ACTIVE}) \equiv \frac{\sum_{i=1}^n (\text{Acc}_i(\text{ACTIVE}) - \text{Acc}_i(\mathbf{Random}))}{\sum_{i=1}^n (\text{OptAcc}_i(\text{ACTIVE}) - \text{Acc}_i(\mathbf{Random}))} \quad (3)$$

Thus, larger values are better (closer to optimal).

4.2 Digit Recognition

The MNIST data set consists of scanned images of handwritten digits 0 through 9. Each image is composed of 28x28 pixels, which are the features for that image. The goal is to learn to distinguish between different digits, allowing for the wide variation that naturally occurs in human handwriting. We used a subset of the full data set that contained 10,000 items, 1,000 from each class (digit). For these experiments, we focus on learning to separate digits 1 and 7, which is one of the more difficult cases. The 8,000 items representing the eight other digits are all unlabelable items, since they are neither 1’s nor 7’s and performance on them

Algorithm	Deff ₃₀₀	95%-trials	95%-speed	Leff ₃₀₀	# unlabelable chosen
Random	1.00	65	1.00	0.00	240 (80%)
Simple	0.61	38	1.71	0.42	284 (95%)
Maxmin	5.06	-	-	-	242 (81%)
Diverse	0.66	34	1.91	0.36	278 (93%)
Prob. Simple	0.69	39	1.67	0.34	274 (91%)
Prob. Maxmin	0.94	68	0.96	0.07	237 (79%)
Prob. Diverse	1.06	-	-	-	241 (80%)
LB-Simple	0.47	38	1.71	0.56	215 (72%)
RU-Simple	0.58	39	1.67	0.45	199 (66%)
LB-Maxmin	4.61	-	-	-	108 (36%)
RU-Maxmin	3.57	-	-	-	130 (43%)
LB-Diverse	0.64	39	1.67	0.39	240 (80%)
RU-Diverse	0.72	55	1.18	0.30	237 (79%)
Prob. LB-Simple	0.57	45	1.44	0.47	217 (72%)
Prob. RU-Simple	0.66	44	1.48	0.37	208 (69%)
Prob. LB-Maxmin	0.98	69	0.94	0.02	127 (42%)
Prob. RU-Maxmin	1.12	-	-	-	115 (38%)
Prob. LB-Diverse	0.95	64	1.02	0.06	239 (80%)
Prob. RU-Diverse	0.94	56	1.16	0.07	240 (80%)
LB-KFF	4.26	-	-	-	91 (30%)

Table 1. Results of experiments on the MNIST data, averaged over 30 runs. The task was to learn to separate digits 1 and 7, with all other digits present as unlabeled data. The best value in each column is in bold. We omitted results for the three middle columns for algorithms that never reached 95% accuracy or performed worse than **Random**.

is irrelevant. For each experiment, we used a subset of 1500 digits for the pool (roughly 150 of each class) and evaluated on a random disjoint set of 1000 test digits (all 1's and 7's, roughly 500 of each). For our SVM we used an RBF kernel with $\gamma = 1$ and $C = 1$.

While this particular problem is contrived, it is very similar to the task of creating a new database similar to MNIST: if the images of digits were extracted using some automatic process, it is highly likely that these examples would sometimes contain items other than digits, such as letters, symbols, or even just noise. An active learning algorithm that learned to be sensitive to these extra classes could speed up the process of labeling these images. We believe that the problem of learning to distinguish between two digits, with other digits thrown in as distractions, is a reasonable analogue to this real problem.

We ran several active learning algorithms on this data set, and for each algorithm we tried both the traditional and the probabilistic variation (**Prob.**), with and without **RU** and **LB**. Table 1 reports for each algorithm, after 300 trials, the deficiency, the number of trials required to reach 95% accuracy, the corresponding 95%-speed (compared to **Random**), the label efficiency, and the number of unlabelable examples chosen.

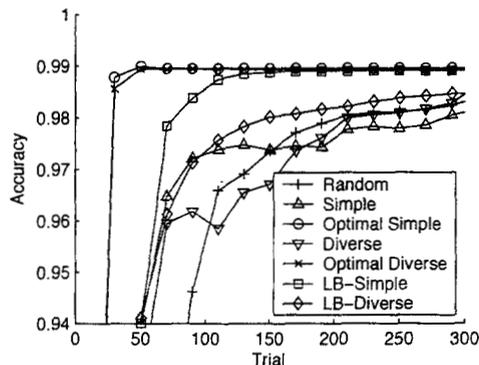


Fig. 3. MNIST results, averaged over 30 runs, for **Simple** and **Diverse**, compared to their optimal performance (no unlabelable examples) and their performance when using **LB**.

LB-Simple achieves the best deficiency and label efficiency, while **Diverse** achieves 95% accuracy in the least number of trials. Interestingly, **LB-KFF** is the most effective approach for minimizing the number of unlabelable examples chosen. However, it has a very poor deficiency score, performing much worse than **Random**. This suggests that strictly minimizing the number of unlabelable examples cannot ensure that accuracy will be maximized. However, KFF is known to perform poorly on most problems; it is specifically designed for a few unusual cases. The overall trend we observe in this table is that **RU** and **LB** almost always improve deficiency and label efficiency. However, they also tend to increase the number of trials required to reach 95% accuracy.

Figure 3 shows the full learning curves for **Simple** and **Diverse**. The optimal performance, when no unlabelable examples are selected, shows the upper bound on performance for each method. Using **LB** significantly improves the performance of each algorithm. (To make all of the graphs in this paper easier to read, we are plotting the mean of the accuracies for 20 neighboring trials.)

4.3 Classifying Splice Junctions in Primate DNA Sequences

As another experiment, we tested our algorithms on the `dna` data set from the StatLog repository [7]. The goal of this task is to use the DNA sequence on either side of a splice junction to distinguish between three classes, corresponding to exon/intron boundaries (EI sites, or “donors”), intron/exon boundaries (IE sites, or “acceptors”), and neither. We focus on the binary subproblem of distinguishing between the two types of splice junctions, in which case we consider the remaining examples to be unlabelable. We assigned the EI sites to the positive class and the IE sites to the negative class. Each feature vector contains 60 DNA base pairs, encoded using three binary features. We trained our SVMs using the same parameters as those used in the one-vs-one experiments by Hsu and Lin [8]: namely an RBF kernel with $\gamma = 2^{-6}$ and $C = 8$. Each experiment was run

Algorithm	Deff ₃₀₀	90%-trials	90%-speed	Leff ₃₀₀	# unlabelable chosen
Random	1.00	125	1.00	0.00	160 (53%)
Simple	0.80	120	1.04	0.19	216 (72%)
Maxmin	1.65	-	-	-	197 (66%)
Diverse	1.23	-	-	-	232 (77%)
Prob. Simple	0.78	112	1.12	0.25	194 (65%)
Prob. Maxmin	0.91	127	0.98	0.13	155 (52%)
Prob. Diverse	1.00	136	0.92	0.00	168 (56%)
LB-Simple	0.30	65	1.92	0.67	106 (35%)
RU-Simple	0.35	72	1.74	0.63	86 (29%)
LB-Maxmin	1.28	-	-	-	31 (10%)
RU-Maxmin	1.01	-	-	-	85 (28%)
LB-Diverse	0.99	142	0.88	0.01	187 (62%)
RU-Diverse	0.84	111	1.13	0.15	160 (53%)
Prob. LB-Simple	0.59	85	1.47	0.47	114 (38%)
Prob. RU-Simple	0.56	85	1.47	0.50	98 (33%)
Prob. LB-Maxmin	0.85	107	1.17	0.23	84 (28%)
Prob. RU-Maxmin	0.70	99	1.26	0.46	78 (26%)
Prob. LB-Diverse	0.94	126	0.99	0.08	158 (53%)
Prob. RU-Diverse	0.94	124	1.01	0.08	157 (52%)

Table 2. Results of experiments on the DNA data, averaged over 30 runs. The task was to learn to separate splice boundaries. Note that LB-KFF is missing, and LB-diverse is using 0.75 instead of 0.5.

on a pool of 1000 examples chosen from the training set of 2000 examples (464 positive, 485 negative, 1051 unlabelable) and tested on the labelable examples from the test set (303 positive, 280 negative).

Table 2 reports on the results with the same set of algorithms as in the previous problem. In this case, the **LB-Simple** algorithm is the clear winner according to all three of our evaluation metrics. For this data set, both the **LB** and **RU** approaches improve the performance of all six active learning algorithm variants in the presence of unlabelable examples. Again, the algorithm that chooses the fewest unlabelable examples, **LB-Maxmin** in this case, is not the one with the best performance. Unlike in the previous experiment, **LB** and **RU** did not increase the number of trials required to reach the cutoff we chose, in this case 90% accuracy.

4.4 Automated Identification of Clouds in MISR Data

The data set shown in Figure 1, which we are making available from our website [9], was collected by the MISR instrument over the Sahara Desert. With the help of an atmospheric scientist, we identified three regions in the image corresponding to clouds, clear land, and dust, and extracted feature vectors for 2,000 pixels from each of those three regions. The feature vector for each pixel consisted of the bidirectional reflectance factor for each pixel and a subset of the

Algorithm	Deff ₃₀₀	95%-trials	95%-speed	Leff ₃₀₀	# unlabelable chosen
Random	1.00	71	1.00	0.00	91 (30%)
Simple	1.63	–	–	–	237 (79%)
Maxmin	7.46	–	–	–	162 (54%)
Diverse	0.82	53	1.34	0.44	217 (72%)
Prob. Simple	0.76	55	1.29	0.36	202 (67%)
Prob. Maxmin	1.03	–	–	–	95 (32%)
Prob. Diverse	0.81	57	1.25	0.44	92 (31%)
LB-Simple	0.87	68	1.04	0.23	112 (37%)
RU-Simple	0.80	74	0.96	0.37	88 (29%)
LB-Maxmin	7.52	–	–	–	35 (12%)
RU-Maxmin	6.33	–	–	–	66 (22%)
LB-Diverse	0.67	52	1.37	0.79	103 (34%)
RU-Diverse	0.67	42	1.69	0.81	87 (29%)
Prob. LB-Simple	0.60	43	1.65	0.59	98 (33%)
Prob. RU-Simple	0.57	41	1.73	0.64	86 (29%)
Prob. LB-Maxmin	1.12	–	–	–	36 (12%)
Prob. RU-Maxmin	1.07	–	–	–	41 (14%)
Prob. LB-Diverse	0.80	57	1.25	0.45	90 (30%)
Prob. RU-Diverse	0.82	66	1.08	0.41	89 (30%)

Table 3. Results of experiments on the MISR data, averaged over 30 runs. The task was to learn to separate cloudy from clear pixels, with dust pixels marked as unlabelable.

pixels within a 5x5 neighborhood, from four different spectral bands and from cameras viewing the scenes from three different angles, yielding 156 real-valued features per example. In all of our experiments, we chose 1000 examples for the training pool and a disjoint set of 1000 examples for the test set. For our SVM we used an RBF kernel with $\gamma = 1$ and $C = 1$.

We have already shown that both the **LB** and **RU** approaches improve the performance of **Simple** on this problem (Figure 2), and Table 3 reports on our complete results for all other algorithms. In this case, the probabilistic variant **Prob. RU-Simple** has the lowest deficiency and best 95%-speed, while **RU-Diverse** has the best label efficiency. Once again, **LB-Maxmin** selects the fewest unlabelable examples but had a poor learning rate.

5 Conclusions and Future Work

Active learning enables the application of machine learning methods to problems where it is difficult or expensive to acquire expert labels. A key barrier to the use of current active learning methods is that it is not always realistic to assume that the expert labeler will be willing (or able) to assign a label to every example. We showed that in some cases, active learning algorithms can actually perform worse than passive learning when a significant class of examples is unlabelable.

We have proposed a new active learning framework where the expert labeler is allowed to decline to label any example, and where the learner attempts to model this unlabelable class to avoid making more queries that return no useful information. We have presented a straightforward method by which any active learning algorithm can be modified to avoid unlabelable examples by training a second classifier to distinguish between the labelable and unlabelable classes. This method should bring us closer to the goal of applying active learning to real-world problems.

In this paper, we have focused on binary classification problems with some unlabelable examples present. In the future, this could be extended to the more difficult problem where the unlabelable examples are precisely those examples that are right on the border between two classes (e.g. very near an SVM's decision hyperplane).

Acknowledgments

The authors wish to thank Dennis DeCoste for suggesting the initial idea that led to this work and for valuable suggestions and feedback along the way. We also thank David Diner, Roger Davies, and Michael Garay for motivating this work by working with us on MISR cloud classification, and Rebecca Castaño and Robert Granat for other valuable feedback.

References

1. Cohn, D., Atlas, L., Ladner, R.: Improving generalization with active learning. *Machine Learning* **15** (1994) 201–221
2. Diner, D.J., Beckert, J.C., Reilly, T.H., Bruegge, C.J., Conel, J.E., Kahn, R.A., Martonchik, J.V., Ackerman, T.P., Davis, R., Gerstl, S.A.W., Gordon, H.R., Muller, J.P., Myneni, R.B., Sellers, P.J., Pinty, B., Verstraete, M.M.: Multi-angle Imaging SpectroRadiometer (MISR) instrument description and experiment overview. *IEEE Transactions on Geoscience and Remote Sensing* **36** (1998) 1072–1087
3. Tong, S., Koller, D.: Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research* **2** (2002) 45–66
4. Cortes, C., Vapnik, V.: Support-vector network. *Machine Learning* **20** (1995) 273–297
5. Brinker, K.: Incorporating diversity in active learning with support vector machines. In: *Proceedings of the Twentieth International Conference on Machine Learning*, Washington, D. C. (2003) 59–66
6. Baram, Y., El-Yaniv, R., Luz, K.: Online choice of active learning algorithms. In: *Proceedings of the Twentieth International Conference on Machine Learning*, Washington, D. C. (2003) 19–26
7. Michie, D., Spiegelhalter, D., Taylor, C.: *Machine Learning, Neural and Statistical Classification*. Prentice Hall, Englewood Cliffs, N.J. (1994) Data available at <http://www.liacc.up.pt/ML/statlog/>.
8. Hsu, C., Lin, C.: A comparison of methods for multi-class support vector machines. *IEEE Transactions on Neural Networks* **13** (2002) 415–425
9. Mazzoni, D.: MISR cloud classification dataset (2004) Available from <http://ml.jpl.nasa.gov/datasets>.