Active Learning with Irrelevant Examples

Dominic Mazzoni, Kiri L. Wagstaff, and Michael C. Burl

Jet Propulsion Laboratory, California Institute of Technology,
Mail Stop 126-347, 4800 Oak Grove Drive, Pasadena CA 91109, USA,
kiri.wagstaff@jpl.nasa.gov

Abstract. Active learning algorithms attempt to accelerate the learning process by requesting labels for the most informative items first. In real-world problems, however, there may exist unlabeled items that are irrelevant to the user's classification goals. Queries about these points slow down learning because they provide no information about the problem of interest. We have observed that when irrelevant items are present, active learning can perform worse than random selection, requiring more time (queries) to achieve the same level of accuracy. Therefore, we propose a novel approach, **Relevance Bias**, in which the active learner combines its default selection heuristic with the output of a simultaneously trained relevance classifier to favor items that are likely to be both informative and relevant. In our experiments on a real-world problem and two benchmark datasets, the Relevance Bias approach significantly improves the learning rate of three different active learning approaches.

1 Introduction

For many classification problems, class labels must be generated by a domain expert and are therefore expensive to acquire. Active learning [1] attempts to reduce this burden by incrementally selecting the most useful items for labeling. Current active learning approaches assume that the expert can provide a valid label for any item in the data set. In this work, we investigate what happens when this assumption is violated due to the presence of *irrelevant* examples (items that cannot be assigned to any of the valid classes). This may occur because the example belongs to an irrelevant class or because it is ambiguous. For example, in the realm of handwritten digit recognition, irrelevant examples include scanning errors, such as smudges or non-digit characters, or truly ambiguous examples, such as a '7' with a very short upper bar that therefore looks like a '1'. Existing active learning methods have no mechanism for handling irrelevant examples.

An obvious strategy for dealing with irrelevant items is to filter the data before training the classifier. While this is a reasonable solution for a benchmark data set, it is unrealistic as a general solution, especially with very large data sets. The process of filtering is as labor-intensive as labeling the entire data set. Since the purpose of active learning is to achieve high performance without requiring that every item be labeled, it is necessary for active learners to be able to work with the unfiltered data.

This paper offers two main contributions. First, we propose an active learning framework where "irrelevant" is a valid response from the expert labeler (Section 2). In this framework, we demonstrate that several popular active learning methods are sufficiently sensitive to irrelevant examples that in some cases they perform worse (that is, learn more slowly) than a random selection (passive) strategy. We also discuss why placing the irrelevant items into a new class and using a multi-class active learning method is ineffective. We then present our second contribution, **Relevance Bias**, a method by which any active learning method can learn to avoid querying irrelevant items (Section 3). Finally, we present experimental results in Section 4 that demonstrate the improvements achieved by active learners with a Relevance Bias and conclude in Section 5.

2 Active Learning and Irrelevant Items

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 $P(x)$. For simplicity, we will discuss active learning in the context of binary classification. In traditional active learning, there exists a classification label $y_i \in \{\pm 1\}$ for each x_i that is available, upon request, from the expert labeler. 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. In each round, the active learner selects an unlabeled item x from $\mathcal{U}$ and receives its label, $y = f(x)$. The learner then updates its classifier based on $\mathcal{L} \cup \{(x, y)\}$. In this section, we describe several active learning methods and then discuss how the active learning problem changes when irrelevant items 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) [2] 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, provides the greatest accuracy improvement. In this work, we consider four data selection methods: three active learning methods and passive (random) selection.

1. **Simple Margin ("Simple")** [3]: Rank each example $x \in \mathcal{U}$ by its distance from the hyperplane and then choose item x with the smallest distance.

2. **MaxMin Margin** [3]: Empirically test which item x will be most effective, in terms of maximizing the separation between the two classes (and therefore minimizing the size of the version space), regardless of which label it receives.
3. **Diverse** [4]: Choose item x that simultaneously minimizes distance to the hyperplane and maximizes the diversity of the new training set, $\mathcal{L} \cup \{(x, y)\}$.
4. **Random**: Choose item x randomly.

Probabilistic Active Learning. Because each of the active learning algorithms described above proceeds heuristically, it will not always be advantageous to select the top-ranked query. Therefore, for each algorithm, we instead used a variant that sorts all of the examples in the pool according to the active learning algorithm's heuristic, and then chooses an item at random from the top p% of the pool instead of choosing the top-ranked example. In our experiments, with $p = 10\%$, we determined that these probabilistic active learners outperformed the "strict" algorithms by 1–6% on all three data sets and never resulted in decreased performance. Therefore, in the remainder of the paper we will report results using probabilistic active learning.

2.2 A Framework for Active Learning with Irrelevant Items

Applying active learners when irrelevant items are present requires a modified learning framework. We model the new expert labeler as a function h that maps items to *three* possible values, $y_i \in \{-1, 0, +1\}$. A value of 0 indicates that the label is irrelevant to the learning task at hand. Again, on each round, the active learner applies a selection function to choose an unlabeled item x from $\mathcal{U}$. The learner proceeds normally unless $h(x) = 0$, in which case it acquires no new information and must wait until the following round to make a new request. For some problems, waiting until the next round can be extremely expensive. For example, we have investigated using active learning to select initial conditions for an asteroid collision simulator [5]. Determining the "label" (outcome) for each set of initial conditions requires running a numerical simulation algorithm for days, and a significant amount of time is lost due to an irrelevant query.

Using this framework, we are able to study each active learning method's response to the presence of irrelevant items. The ideal active learner would avoid the irrelevant items completely, since they cannot help improve the margin or reduce the size of the feature space. However, if the active learner's heuristic for item selection coincides with the kind of irrelevant examples that are present, the learner may devote the majority of its queries to the irrelevant items. In Section 4, we will show experimentally that this problem is significant enough that it can cause active learning methods to perform worse than random selection. In keeping with the classification goals, we evaluate performance on the relevant items only.

An intuitive approach to dealing with irrelevant items would be to use a multi-class active learner and provide it with three classes: the positive, negative, and irrelevant items. However, this approach requires that the active learner

```
SELECT-RB(active learner $\mathcal{A}$, relevance classifier $C^*$)
1. Rank all items $x$ in the unlabeled pool $\mathcal{U}$ using $\mathcal{A}$.
2. Normalize all $\mathcal{A}(x)$ scores into the range $[0, 1]$.
3. Calculate $P(\text{relevant}|x)$ using $C^*$.
4. Select $x$ such that $\mathcal{A}(x) \times P(\text{relevant}|x)$ is maximized, and request $y = h(x)$,
the label for $x$.
5. If $y = 0$ (irrelevant), add $x$ to $\mathcal{R}$; otherwise, add $x$ to $\mathcal{L}$. Re-train $C^*$.
```

Fig. 1. Pseudo-code for adding a Relevance Bias to active learner $\mathcal{A}$.

devote resources (queries) to accurately modeling the irrelevant class, which detracts from the real classification goal. In the next section, we introduce our approach, which cleanly separates the goals of (1) high performance on the relevant classes and (2) minimizing the number of irrelevant queries.

3 Solution: Active Learning Relevance Bias

We propose an active learner that, in addition to selecting the most informative items for labeling, also learns to separate relevant from irrelevant items. This approach can be adopted by any existing active learning method. The new active learner collects irrelevant items x in a set $\mathcal{R}$ of rejected queries, then 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 active learner uses this relevance classifier, C^*, to influence its decision about which example should be chosen next. Classifier C^* is unlikely to be 100% accurate, especially in early rounds, so relying on it to strictly filter the pool $\mathcal{U}$ may not yield the best results. Instead, we use C^* to influence the active learner's *rankings* of items in the pool, creating a **Relevance Bias (RB)**. Figure 1 outlines pseudocode that replaces a normal active learner's item selection method. In step 4, the learner combines its ranking of the items with the probability that they are relevant to yield a final decision about which item to query. In our experiments, using C^* as a modifier rather than a filter increased performance by up to 30%.

We trained an SVM for C^* using the same parameters (kernel function and regularization parameter) as were used for $\mathcal{A}$. However, any learning method that outputs a probability can be used. When C^* is an SVM, $P(\text{relevant}|x)$ can be approximated in several ways, such as clipping values outside the range $[-1, 1]$ and mapping them to the range $[0, 1]$, or using Platt's technique of mapping the SVM outputs to a sigmoid probability model [6]. We used the former method in this work.

4 Experimental Results

To evaluate the effectiveness of our proposed approach, we conducted experiments on a real-world data set and two benchmark problems. Key data set

Table 1. Summary of the data sets, including the SVM parameters used, the choice of positive, negative, and irrelevant classes, and number of items in each class. All experiments used an RBF kernel.

	Number of	Training set (# items)			SVM params	
Data set	features	Positive	Negative	Irrelevant	γ	C
MISR	156	cloudy (100)	clear (100)	dusty (100)	1.0	1.0
DNA	180	EI (50)	IE (50)	neither (50)	2^{-6}	8.0
MNIST	784	‘1’ (100)	‘7’ (100)	others (800)	1.0	1.0

characteristics, including SVM parameters, are shown in Table 1. The disjoint test sets are composed solely of relevant items.

MISR. This data set consists of the pixels in an image that was collected by the Multi-angle Imaging SpectroRadiometer (MISR) instrument in Earth orbit over the Sahara Desert on February 6, 2004. The goal is to build a classifier that distinguishes cloudy and clear pixels. This particular image also contains several dusty pixels, which fall into neither class and are therefore considered irrelevant. For each pixel, we extracted 156 features: the bidirectional reflectance factor for the pixel and a subset of the pixels within a 5x5 neighborhood, from four different spectral bands and from cameras viewing the scenes from three different angles. We selected 300 pixels randomly from the image to form the training set, 100 each of the cloudy, clear, and dusty classes.

DNA. This is the `dna` data set from the StatLog repository [7]. The goal is to use the DNA sequence on either side of a splice junction to distinguish between exon/intron boundaries (EI sites, or “donors”) and intron/exon boundaries (IE sites, or “acceptors”). Some splice junctions fall into neither category (irrelevant). Each feature vector consists of 180 features (60 DNA base pairs, each encoded using three binary features). We ran experiments on 150 examples chosen randomly from the training set of 2000 examples, using the SVM hyperparameters as in Hsu and Lin’s one-vs-one experiments [8].

MNIST. This data set consists of scanned images of handwritten digits [9]. Each image is composed of 28x28 pixels. We used a subset of the full data set that contained 1000 items, 100 from each class (digit). For these experiments, we focus on learning to distinguish between digits 1 and 7, which is one of the more difficult cases. The 800 items representing the eight other digits are all irrelevant items, since they are neither 1’s nor 7’s.

4.1 Active Learning Results

We performed an empirical comparison for all three active learning methods against random selection and the RB-enhanced versions of each method. In addition to plotting learning curves, we also compute a scalar value that captures overall performance, AUC (area under the curve). The AUC of algorithm $\mathcal{A}$ after n rounds is the sum of the accuracy it obtained at each round from 1 to n,

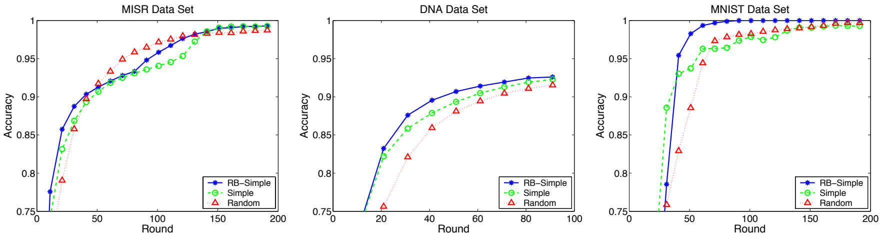

Fig. 2. Active learning (Simple) with and without a Relevance Bias (100 trials).

normalized by n.

$$\mathrm{AUC}_n(\mathcal{A}) = \frac{1}{n}\sum_{t=1}^{n} \mathrm{Acc}_t(\mathcal{A}) \tag{1}$$

If n is clear, we omit the subscript. In general, a higher AUC indicates faster, more efficient learning.

Figure 2 shows the behavior of Simple, RB-Simple, and Random on each data set (the other active learners are omitted from the figures to reduce clutter). The quantified AUCs observed for all three active learners are shown in Table 2. We observe several cases where regular active learning performs worse than random selection. The MaxMin algorithm is particularly sensitive to the presence of irrelevant items in the MISR and MNIST data sets. Overall, we find that RB-Diverse yields the best performance.

With the sole exception of Simple's performance on the MNIST data set, adding a Relevance Bias to each method improves performance (AUC), sometimes dramatically. To more clearly illustrate this phenomenon, Figure 3 (top) shows the (cumulative) number of irrelevant items that each algorithm selected. We can see that Simple has a definite bias towards selecting irrelevant items (more than expected by random chance). In contrast, RB-Simple chooses far fewer irrelevant items and correspondingly achieves higher performance.

Because the performance of an RB-enhanced active learner is dependent on the accuracy of its C^* classifier, we also examined the C^* learning curves. Figure 3 (bottom) shows these curves for Simple, RB-Simple, and Random (evaluation is over a separate set of items not included in $\mathcal{U}$). C^* quickly achieves a high

Table 2. AUC for three active learners compared to Random, the RB-enhanced versions, and Multi-class active learning (100 trials). The best result for each data set is in bold, and results that are significantly worse than Random (95% confidence level) are italicized.

			Simple		MaxMin		Diverse		
Data set	# Rounds	Random	Regular	RB	Regular	RB	Regular	RB	Multi-class
MISR	200	90.6	90.3	91.3	*65.5*	*76.7*	90.7	**92.0**	*86.8*
DNA	100	79.3	81.7	**82.8**	78.1	78.8	81.7	**82.8**	*73.5*
MNIST	200	87.1	88.0	88.0	*54.0*	*65.0*	89.4	**90.5**	*73.2*

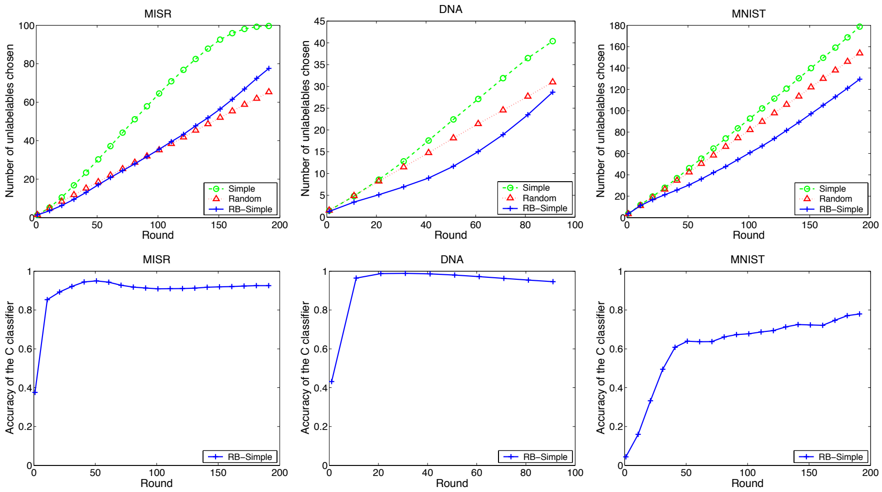

Fig. 3. Top: Comparison between Simple, Random, and RB-Simple in terms of number of irrelevant examples queried; lower values are better. Bottom: Learning curves for RB-Simple's C^* (relevance) classifier. All results are averaged over 100 trials.

level of performance on the MISR and DNA data sets. However, C^* learns more slowly on the more complex MNIST problem (80% of the data set is irrelevant), explaining why RB-Simple's AUC is not an improvement over Simple.

We also tested the multi-class approach (Section 2.2) for each of our data sets. In each case, we trained a multi-class active learner [10] to discriminate between the positive, negative, and irrelevant classes. As shown in Table 2, the multi-class active learner also performs worse (learns more slowly) than Random, because it aims to simultaneously maximize performance over all three classes and must therefore devote a large number of queries to the third (irrelevant) class. In contrast, the Relevance Bias approach can provide performance gains even when C^* is not yet fully accurate, because the majority of its effort is devoted to modeling the two critical classes. This intuition is confirmed experimentally: we find that RB-Simple strongly outperforms the multi-class learner, by 5–19%.

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. Real-world data sets may contain items that are irrelevant to the user's classification goals. When filtering these items is not a realistic option, it is essential that active learning methods be able to cope with the presence of irrelevant items. However, existing active learning algorithms can perform worse than passive learning in this situation.

We have proposed a new active learning framework that allows for any item to be assigned to an "irrelevant" class. We presented a novel method, Relevance Bias, by which any active learning algorithm can be modified to avoid irrelevant examples by training a second classifier to distinguish between the relevant and irrelevant items. This method consistently improves the performance of active learners when irrelevant items are present. We have also shown that the multi-class approach does not perform as well as the Relevance Bias approach.

An important extension to this work will be to add a mechanism that compensates for the fact that C^* cannot perfectly distinguish relevant from irrelevant items (especially early on). For example, the algorithm could estimate C^*'s accuracy via leave-one-out cross-validation and adjust the strength of the relevance bias over time. Thus, initial rounds will be more exploratory, while later ones can rely more strongly on C^*'s recommendations.

Acknowledgments. We thank Dennis DeCoste for suggesting the initial idea for this work and for valuable suggestions and feedback along the way. We also thank David Diner, Roger Davies, and Michael Garay for working with us on MISR cloud classification, and Rebecca Castaño and Robert Granat for other valuable feedback. The research described in this paper was funded by the NASA AIST Program and was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

References

1. Cohn, D., Atlas, L., Ladner, R.: Improving generalization with active learning. Machine Learning **15**(2) (1994) 201–221
2. Cortes, C., Vapnik, V.: Support-vector network. Machine Learning **20** (1995) 273–297
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. 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
5. Burl, M.C., DeCoste, D., Enke, B., Mazzoni, D., Merline, W.J., Scharenbroich, L.: Automated knowledge discovery from simulators. In: Proceedings of the Sixth SIAM International Conference on Data Mining. (2006)
6. Platt, J.C.: Probabilities for SV Machines. In Smola, A.J., Bartlett, P., Schölkopf, B., Schuurmans, D., eds.: Advances in Large Margin Classifiers, MIT Press (1999) 61–74
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. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11) (1998) 2278–2324
10. Tong, S.: Active Learning: Theory and Applications. PhD thesis, Stanford University (2001)