To Re(label), or Not To Re(label)

Christopher H. Lin
University of Washington
Seattle, WA
chrislin@cs.washington.edu

Mausam
Indian Institute of Technology
Delhi, India
mausam@cse.iitd.ac.in

Daniel S. Weld
University of Washington
Seattle, WA
weld@cs.washington.edu

Abstract

One of the most popular uses of crowdsourcing is to provide training data for supervised machine learning algorithms. Since human annotators often make errors, requesters commonly ask multiple workers to label each example. But is this strategy always the most cost effective use of crowdsourced workers? We argue "No" — often classifiers can achieve higher accuracies when trained with noisy "unilabeled" data. However, in some cases relabeling is extremely important. We discuss three factors that may make relabeling an effective strategy: classifier expressiveness, worker accuracy, and budget.

Introduction

Data annotation is one of the most common crowdsourcing applications on labor markets, such as Mechanical Turk, as well as on internal crowdsourcing platforms at companies like Microsoft and Google. Since human workers are error prone, requesters commonly ask multiple workers to redundantly label each example, because multiple workers can simulate an expert worker (Snow et al. 2008). Highly accurate labels may be inferred from the multiple labels, either using a policy such as "Ask two workers and recruit a third to break ties if needed" or more complex EM-style approaches (Dawid and Skene 1979; Whitehill et al. 2009; Lin, Mausam, and Weld 2012b). Indeed, some researchers have developed extremely sophisticated algorithms to guide the relabeling process (Dai et al. 2013; Wauthier and Jordan 2011; Ipeirotis et al. 2013). But a fundamental question remains unaddressed: "Is it better to spend an incremental dollar asking a worker to relabel an existing example or to label a new example?"

In some cases relabeling is clearly necessary. For example, if the data is to be used as a test set, then high accuracy is paramount. More often, however, the data is being annotated in order to *train* a learning algorithm. In this case, whether the resulting learned classifier will have higher accuracy when trained on m noisy examples instead of on a smaller set of (say $m/3$) examples with more accurate labels is unclear.

We first set out to answer this question by considering a subset of real-world datasets from the UCI Machine Learning Repository (Bache and Lichman 2013). We simulated a noisy annotation process with a fixed budget using a simple, deterministic vote aggregation scheme. Specifically, we considered an optimized majority voting scheme, which we call *j/k relabeling*, where we ask up to k workers to label an example, stopping as soon as $j = \lceil k/2 \rceil$ identical responses are received. *Unilabeling* refers to the strategy that only collects one label per example, i.e., 1/1 relabeling. Figure 1 shows the ratio of the accuracy achieved by classifiers trained using relabeling (denoted as *relabeling accuracy*) to the accuracy achieved by classifiers trained using unilabeling (denoted as *unilabeling accuracy*). The results were inconclusive and puzzling. Relabeling helped in 5 of 12 domains, but it was unclear when relabeling would be useful and what level of redundancy would lead to the best results.

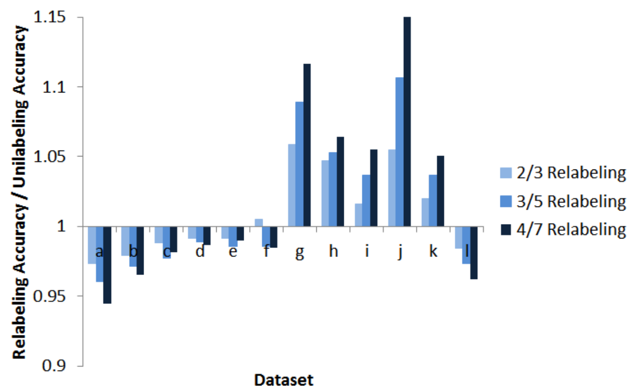

Figure 1: Relabeling training data improved learned-classifier accuracy on 5 out of 12 real-world domains (see Table 1) when workers were highly noisy ($p = 0.55$). But why these domains and not the others?

To understand this phenomenon, we identified three characteristics of learning problems that might affect the relative relabeling versus unilabeling performances. These three dimensions include (1) the inductive bias of the learning algorithm, (2) the accuracy of workers, and (3) the budget.

In the rest of this paper we seek to study the effect of each of these dimensions on relabeling power. We expect to find general principles that will help us determine whether to unilabel or relabel for a given problem.

Problem Setting

Since we address complex issues and this study is the first of its kind, we make a number of simplifying assumptions. We focus on binary classification problems with symmetric loss functions. We assume that workers have uniform (but unknown) error rates. Finally, this paper focuses on passive learning, where the next data point is uniformly sampled from an unlabeled dataset, as opposed to actively picked based on current classifier performance. While we believe that the insights drawn from our setup will also inform other settings, such as active learning or pools of workers with varying abilities, we leave these extensions to future work.

Let $\mathcal{X}$ denote the space of examples and D, its distribution. We consider binary classification problems of the following form. Given some hypothesis class, $\mathcal{C}$, which contains a set of mappings from $\mathcal{X}$ to $\{0, 1\}$, and a budget b, the goal is to learn some true *concept* $c : \mathcal{X} \mapsto \{0, 1\}$ by outputting the hypothesis $h \in \mathcal{C}$ which minimizes the expected error $\epsilon = P_{x \sim D} c(x) \neq h(x)$. Asking a worker to label an example incurs a fixed unit cost of 1. We assume that each worker exhibits the same (but unknown) accuracy $p \in (0.5, 1]$, an assumption known as the *classification noise model* (Angluin and Laird 1988), and we assume worker errors are independent.

Let $\zeta_k : \{0, 1\}^k \mapsto \{0, 1\}$ denote the *aggregation function* that produces a single aggregate label given k multiple labels. The aggregation function represents how we consolidate the multiple labels we receive from workers. Since all workers are equally accurate in our model, majority voting, a common aggregation function that takes k votes and outputs the class that received greater than $k/2$ votes, is an effective strategy. In all our experiments we use a simple optimization of majority vote, j/k relabeling, and stop requesting votes as soon as a class obtains $j = \lceil k/2 \rceil$ votes.

We also define $\eta_{\zeta_k} : [0, 1] \to [0, 1]$ as the function that, given the number of labels k that will be aggregated by ζ_k and the accuracy p of those labels, calculates the probability that the answer returned by the aggregation function ζ_k will be correct. In other words, it outputs the *aggregate accuracy*, the probability that the aggregate label will be equal to the true label ((Sheng, Provost, and Ipeirotis 2008; Ipeirotis et al. 2013) call this probability *integrated quality*). As we will see, η is an important function that characterizes the power of relabeling.

In order to train the best classifier possible, an intermediate goal, and our goal in this paper, is to determine the scenarios under which relabeling is better than unilabeling, assigning a single worker to annotate each example. In other words, given the dataset and classifier, we would like to determine whether or not examples should be relabeled (and with what redundancy) in order to maximize the classifier's accuracy.

The Effect of Inductive Bias

We first consider how the inductive bias of a classifier affects the relative performance of relabeling and unilabeling. We primarily use two tools to address the question: a theoretical analysis of error bounds using PAC-learnability and a series of empirical experiments on simulated data. However, we first provide some intuition.

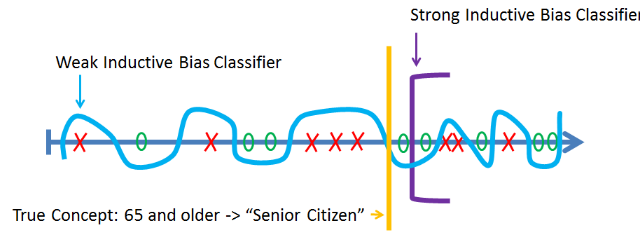

Figure 2: An "O" represents an example labeled as a "senior citizen" and an "X" designates an example labeled as "not a senior citizen." The purple threshold represents the correct classification of all people older than 65 as senior citizens.

Recall that a classifier's inductive bias characterizes the range of hypotheses that it will consider. A weakly regularized learner willing to consider a vast number of hypotheses that can be represented with an expressive hypothesis language, e.g. decision tree induction, is said to have a *weak inductive bias*. In contrast, logistic regression, which learns a linear model, has a stronger inductive bias. Why might relabeling effectiveness depend on the strength of a classifier's inductive bias?

As classifier bias gets weaker, its ability to fit training data patterns increases and the likelihood that it overfits increases. Consider the effect of noise on overfitting. As the noise in training labels increases, overfitting starts to hurt more, because the classifier not only overfits data, but also overfits *the wrong data*. Therefore, we predict that with weaker inductive bias, relabeling will become more necessary to prevent large overfitting errors.

As an illustration, suppose we want to classify whether or not a person is a "senior citizen," based on his/her age. Let the instance space $\mathcal{X}$ be people between the ages of 0 and 100 and the distribution D uniform. Let us suppose the target concept is the simple threshold that everyone older than 65 is a senior citizen. For a hypothesis class $\mathcal{H}_1$ that consists of all thresholds on $[0, \infty]$, the VC dimension is 2 (strong inductive bias). This hypothesis class is quite robust to noise, as shown in Figure 2. As long as the labeling accuracy is above 50%, a learning algorithm using $\mathcal{H}_1$ will probably get the threshold approximately correct, so relabeling is not really necessary. Furthermore, spending one's budget on additional examples increases the chance of getting examples that are close to the 65 year boundary and facilitates an accurate hypothesis.

Now consider a hypothesis class $\mathcal{H}_2$ that allows the classifier to arbitrarily subdivide the space into regions (as in a decision tree with unbounded depth). This hypothesis class is extremely expressive and its VC dimension is equal to the size of X (weak inductive bias). Given the set of noisy examples in Figure 2, a learning algorithm using $\mathcal{H}_2$ is very likely to overfit and achieve low accuracy. In this case, relabeling is very important, because it reduces the likelihood that the classifier overfits the noise.

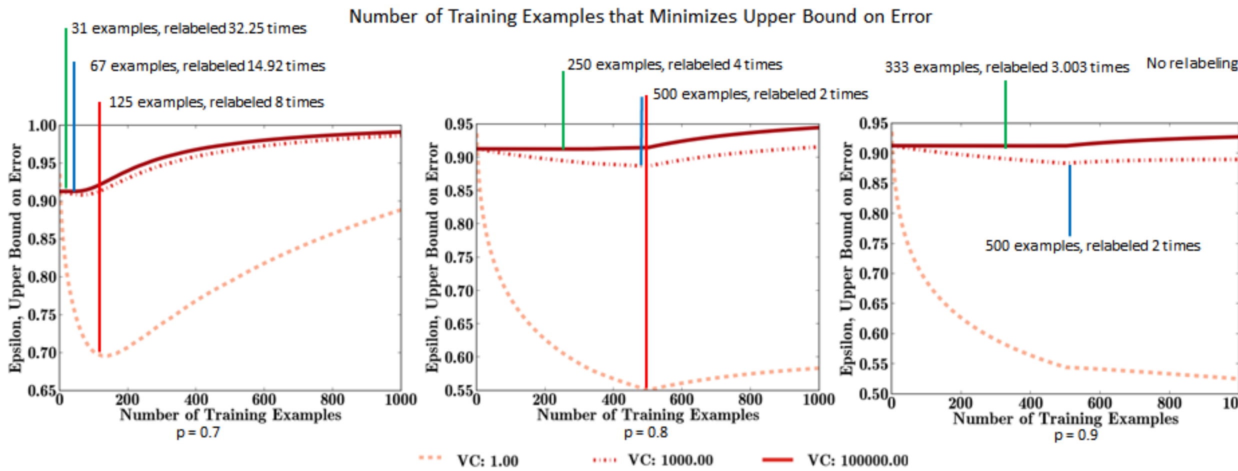

Figure 3: Given a fixed budget of 1000, we see in various settings of worker accuracy that as the VC dimension increases, more relabeling of a fewer number of examples achieves lower upper bounds on classification error.

Bound Analysis

We now make these intuitions precise by bounding classification accuracy in terms of the classifier's Vapnik-Chervonenkis (VC) dimension (Vapnik and Chervonenkis 1971). Recall that the VC dimension of a classifier measures the size of the largest finite subset of $\mathcal{X}$ that it is capable of classifying correctly (shatter). A classifier may make errors when trying to learn datasets with size larger than its VC dimension, but is guaranteed to have a hypothesis that can distinguish all power sets of a dataset with size less than or equal to its VC dimension. A higher VC dimension corresponds to weaker inductive bias.

We assume that the concept class we are considering, $\mathcal{C}$, is Statistical-Query (SQ) learnable (Kearns 1993). While the theory of SQ-learnability is beyond the scope of this paper, this assumption basically guarantees the existence of a classifier that can PAC-learn the target concept under noise. Aslam & Decatur (1998) provide a sufficient bound on m, the number of samples needed to learn to a given accuracy. If each sample is incorrectly labeled with probability at most ξ_0, then an error less than ϵ with probability at least $1-\delta$ is guaranteed if m satisfies:

$$m = \mathcal{O}\Bigg(\frac{1}{\tau^2\epsilon^2(1-2\xi_0)^2}\log^2\frac{1}{\epsilon}\Bigg((VC + VC\log\frac{1}{\epsilon}\log\log\frac{1}{\epsilon})\cdot \log(\frac{1}{\tau\epsilon(1-2\xi_0)}\log\frac{1}{\epsilon}) + \log\frac{1}{\delta}\Bigg)\Bigg).$$

where τ is a parameter that controls how easily the problem is SQ-learnable.

Notice that in our problem setting, the noise rate ξ_0 depends on the total budget b, the accuracy of the workers p, and the number of examples m used to train the classifier. Therefore, by fixing all parameters except for m and ϵ, we can use this bound to find the $m \leq b$ that minimizes ϵ. We now analyze this bound when the aggregation function ζ is a majority vote.

We begin by ensuring that the resulting curve is smooth. Since m may not divide the budget b perfectly, one or more training examples may get an additional label. Let $K = \lfloor\frac{b}{m}\rfloor$ be the minimum number of labels each example has if we train using m examples with budget b. Let $m_{K+1} = b - (mK)$ denote the number of examples with $K+1$ labels. Let $m_K = m - m_{K+1}$ be the number of examples with K labels. Then, we compute ξ_0 as:

$$\xi_0 = m_K(1.0 - \eta_{\zeta_K}(p)) + m_{K+1}(1.0 - \eta_{\zeta_{K+1}}(p))$$

Because the function that computes aggregate accuracy for majority vote, η_{ζ_k}, is not defined for k that is even, we use the defined points for when k is odd along with their reflections across the axes and fit a logistic curve of the form

$$\eta_{\zeta_k} = \frac{c_0}{1+e^{-c_1(k-c_2)}}$$

in order to estimate η_{ζ_k}. We use Scipy's curve fitting library (Jones et al. 2001), which uses the Levenberg-Marquardt algorithm. The resulting curve is not perfect, but accurately reflects the shape of η_{ζ_k}.

Solving for ϵ analytically is difficult, so we employ numerical methods. Since we are only concerned with the change in ϵ as a function of m across various VC dimensions, constant factors don't matter so we set $\tau = 0.1$ and $\delta = 0.1$. Now we use Scipy's optimization library (Jones et al. 2001) to solve for the error bound, ϵ, for all values of $m < b = 1000$.

Figure 3 shows the curves that result for various settings of worker accuracy and VC dimension. We see that as the VC dimension increases, m, the number of examples (each labeled $\approx 1000/m$ times) that minimizes the upper bound on error, decreases. Thus the optimal relabeling redundancy increases with increasing VC dimension. Furthermore, note that when workers are 90% accurate, unilabeling yields the lowest bounds at low VC dimensions while relabeling produces the lowest bounds when VC dimension is high.

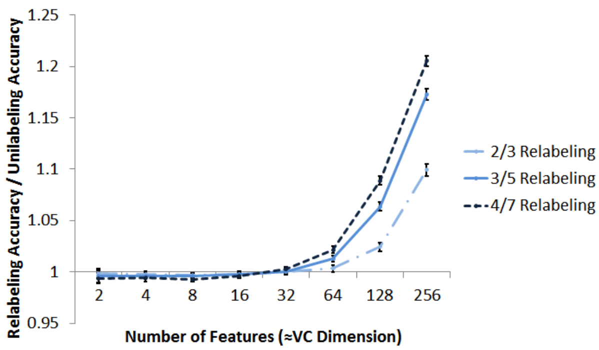

Figure 4: As the number of features (VC dimension) increases, relabeling becomes more and more effective at training an accurate logistic regression classifier.

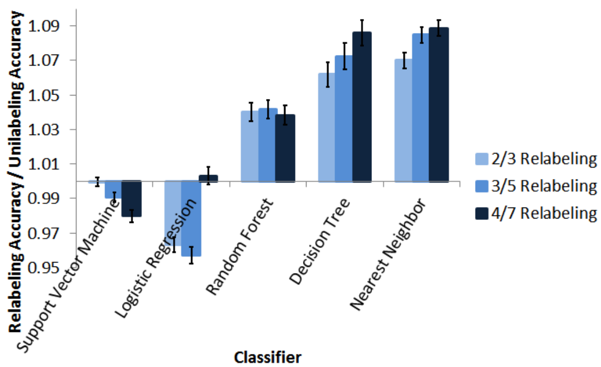

Figure 5: Classifiers with weaker inductive bias tend to benefit more from relabeling.

Our analysis suggests a method to pick *label redundancy*, the number of times to relabel each example. We can simply choose the value of m that minimizes the upper bound on error for the given VC dimension. One caveat: the value which produces the minimum error *bound* does not necessarily guarantee the minimum *error*, unless the bound is tight. Still, we plan to experiment with this heuristic in the future.

Simulated Datasets

We now present experiments that empirically study the effect of inductive bias on relabeling accuracy. For these experiments we test on an artificial dataset, which allows us to control for various parameters. Our datasets contain two Gaussian clusters, which correspond to the two classes. To generate a dataset, we first we pick the number of features to be $l = 50$. Then we randomly pick two means, $\mu_1, \mu_2 \in [0,1]^l$. Next we randomly pick two corresponding covariance matrices $\Sigma_1, \Sigma_2 \in [0,1]^{l \times l}$.

For each Gaussian cluster (class), we generate an equal number of examples. Setting a labeling budget of $b = 500$, we can now train classifiers. We compare a unilabeling strategy against relabeling strategies using $2/3$-, $3/5$- and $4/7$-relabeling. We simulate moderately accurate workers

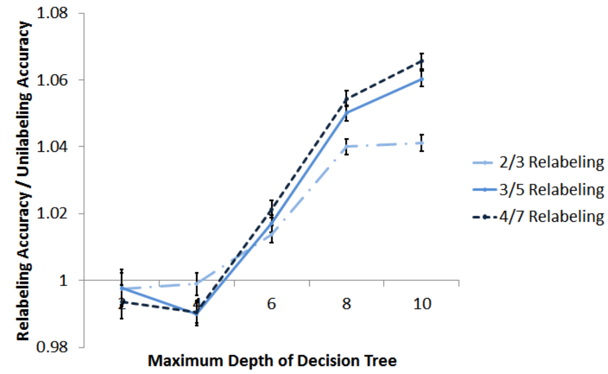

Figure 6: Decision trees with lower regularization tend to benefit more from relabeling.

($p = 0.75$) using the classification noise model. We average each strategy over 1000 runs, and use standard error to compute 95% confidence intervals.

Figure 4 shows how VC dimension affects the power of relabeling. Here we use a logistic regression classifier and set an $l2-$regularization with a balanced regularization constant such that relabeling strategies (which receive a fewer number of examples) are not unfairly over-regularized. We use a linear classifier so that as we vary the number of features, we vary the VC dimension, which is equal to the number of features plus one (Dudley 1978). We see that as the VC dimension increases, relabeling becomes more cost effective.

Figure 5 shows how different types of classifiers perform. We use classifiers from the Scikit-learn (Pedregosa et al. 2011) package in their default settings. We see that logistic regression and support vector machine (SVM) both perform best with a unilabeling strategy, but decision trees, random forests, and nearest neighbor classifiers do not, because these classifiers have high expressiveness and weak inductive bias.

Figure 6 shows how a decision tree classifier performs as we vary its maximum depth. Since increasing the depth of a decision tree increases the expressiveness of the corresponding logical formula, increasing depth corresponds to weaker inductive bias. We see that as the maximum depth increases, relabeling becomes the more effective strategy.

Our experiments validate the insights described previously. Overall, we believe that for low-dimensional data and strongly biased classifiers, unilabeling may be the method of choice. If, however, the VC dimension is high, then relabeling is likely preferred.

The Effect of Worker Accuracy

We now consider the effect of worker accuracy on relabeling. In the extreme case, relabeling cannot possibly help if workers are perfect. Conversely, there is great potential to improve the quality of one's training set when worker accuracies are barely over $p = 0.5$. Hence, our a priori belief was that relabeling should be more effective when workers are less accurate.

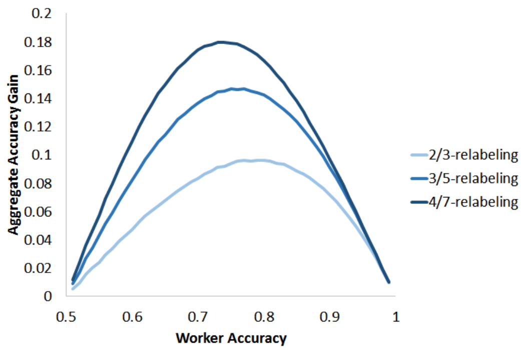

Figure 7: The increase in the aggregate accuracy of the training set data when using relabeling instead of unilabeling for various worker accuracies.

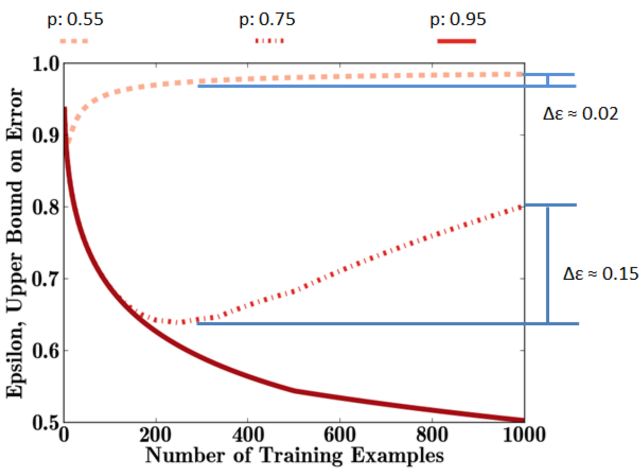

Figure 8: When the workers are moderately accurate ($p = 0.7$), the difference in the upper bound on error due to typical relabeling strategies is greatest.

However, this intuition is faulty as we now explain. Indeed, (Ipeirotis et al. 2013) have shown that typical relabeling strategies have the maximum effect *on the accuracy of a training set* (not on the resulting classifier), when workers are of intermediate abilities. Consider Figure 7, which plots the increase in aggregate accuracy of the training data when relabeling instead of unilabeling as a function of worker accuracy. The three peaks happen between 0.73–0.79. We also observe that 2/3-relabeling only improves accuracy by about 0.1 in the best case, whereas 4/7-relabeling can get to an almost 0.2 increase. Furthermore, as the amount of relabeling is increased, the peak in accuracy gain moves to the left, suggesting that strategies with increasing amounts of relabeling have their maximum effect as the workers become less accurate.

But these past results only apply to the quality of a training set, not the accuracy of the resulting classifier. By considering the accuracy of the classifier, we must address the confounding factor that eschewing relabeling frees budget to be spent labeling new examples. To study this scenario further we continue the analysis technique from the previous section to produce Figure 8, which compares various upper bound curves for different settings of worker accuracy in the setting of VC=1 and budget = 1000. Consider the difference in error bound as m ranges between 333 (when every example is labeled 3 times) to 1000 (when thrice as many examples are labeled once). This delta is much greater when the workers are moderately accurate ($p = 0.75$) than for other settings of worker skill. We see similar patterns for other settings of labeling redundancy and VC dimension. These differences in error bound support the belief that typical relabeling strategies are most likely to reduce classifier error when p is not an extreme value.

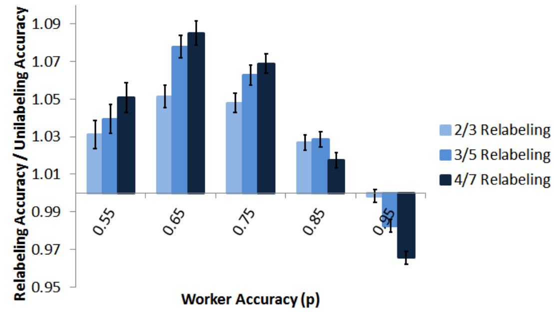

Figure 9: For simulated Gaussian datasets, relabeling strategies based on majority-vote are most powerful at moderate values of worker accuracy.

Simulated Datasets

To confirm these insights, we again present experimental analysis using our artificial Gaussian datasets, and use varying settings of worker accuracy. As in the previous section, we fix the number of features to be $l = 50$, and the budget to be $b = 500$. We train using decision trees and set the maximum depth to be 10. For this experiment, instead of averaging over 1000 runs, we average over 2000 runs in order to create tight confidence intervals across varying worker accuracies.

Figure 9 shows our results. The more highly redundant approaches, 4/7- and 3/5- relabeling, clearly have their maximum benefit when workers are 65% accurate. On the other hand, 2/3-relabeling has its maximum benefit somewhere between $p = 0.65$ and $p = 0.75$. These results (and similar ones that we find using different classifiers like logistic regression) mirror our intuition and our theoretical analysis. Thus, choosing the correct amount of relabeling redundancy is a complex decision which ideally should be informed by knowledge of worker accuracy.

The Effect of Budget

We now investigate the effect of budget on relabeling power. Intuitively, one might think that as the budget increases, relabeling will become the more effective strategy, because in the extreme case of when the budget is infinitely large, we should clearly label each example infinitely many times. Such a strategy allows us to train the classifier using the en-

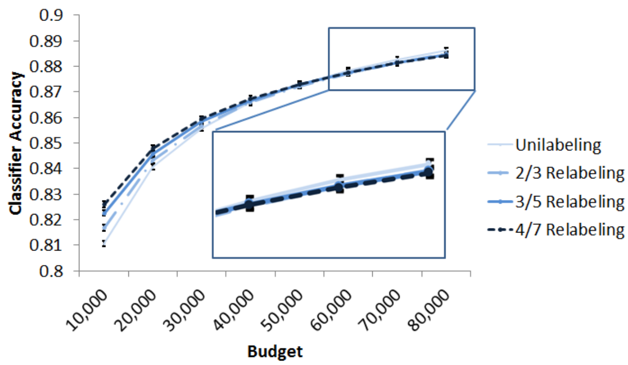

Figure 10: When $p = 0.75$, relabeling strategies initially achieve higher accuracies than unilabeling, but are eventually defeated.

tire set of noiseless examples. However, this extreme case does not arise in typical finite budgets, and in fact our experiments show quite an opposite trend.

We again train a decision tree with a maximum depth of 10 using our simulated Gaussian datasets with $l = 50$ features and moderately accurate workers ($p = 0.75$), and we vary the total budget. We plot the resulting learning curves in Figure 10, which are averaged over 1000 runs. We see that while initially relabeling strategies achieve higher accuracy than unilabeling, eventually unilabeling overtakes relabeling somewhere around a budget of 60000, because the slope of the unilabeling learning curve is higher than that of the relabeling curves. Indeed, increasing the amount of relabeling decreases the slope while increasing the initial accuracy.

Upon reflection, such a result makes sense. With very low budgets, classifiers are unable to generalize well with noisy data. However, with a large enough budget, the noise becomes simply that: irrelevant noise. There are enough accurate examples such that the classifier can learn a correct hypothesis that ignores the noisy data.

We also plot learning curves using poor workers ($p = 0.55$) in Figure 11 in order to show the effect more clearly. When the budget reaches approximately 4000, unilabeling begins to achieve higher accuracies than relabeling. Interestingly, these two figures also show the effect of worker accuracy. Unilabeling takes much longer to start achieving higher accuracies when the workers are moderately accurate because relabeling strategies are most powerful in this setting.

We do not show a figure in the case when the workers are excellent, but note that in this case, unilabeling is strictly better than relabeling strategies. This result again intuitively makes sense, as there is no need to improve labels when the labels need not any improvement. Finally, we note that we see similar trends with other classifiers (*e.g.*, logistic regression, random forests) as well. We conclude that increasing the budget tends to benefit unilabeling, and the point at which unilabeling defeats relabeling is controlled by other factors, like worker accuracy.

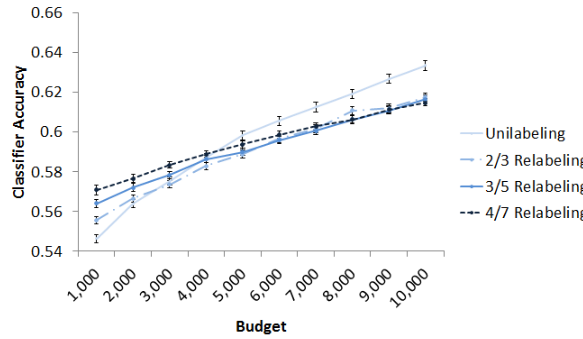

Figure 11: When $p = 0.55$, relabeling strategies initially achieve higher accuracies than unilabeling, but are defeated earlier than when $p = 0.75$.

Real Dataset Experiments

Dataset	# Features	# Examples
(a) Breast Cancer	9	699
(b) Bank Note Authentication	4	1372
(c) Seismic Bumps	18	2584
(d) EEG Eye State	14	14980
(e) Sonar	60	208
(f) Breast Cancer Diagnostic	30	569
(g) Hill-Valley	100	606
(h) Hill-Valley with Noise	100	606
(i) Internet Ads	1558	2359
(j) Gisette	5000	6000
(k) Farm Ads	54877	4143
(l) Spambase	57	4601

Table 1: The 12 datasets we use, with the total number of examples and number of features in each.

We now compare unilabeling and relabeling using 12 datasets from the UCI Machine Learning Repository (Bache and Lichman 2013), with the goal of matching trends observed in simulated datasets with real-world datasets. We list the datasets in Table 1. We use half of the available examples as the budget, and hold out 15% of the examples for testing. We simulate workers at accuracies of $p = 0.55$ and $p = 0.75$. We train a logistic regression classifier (a linear classifier) so that we can observe trends with varying VC dimension. Our results, averaged over 1000 runs, are shown in Figures 1 and 12.[1]

We see that when the workers are poor ($p = 0.55$), domains are split evenly with respect to which strategy is performing better. This can be explained based on the VC dimension. In the five domains with high VC dimensions (100 or more features) relabeling is outperforming unilabeling.

[1]Under the stated parameters, when training using an expressive classifier, like a decision tree, relabeling tends to always perform better than unilabeling. However, the achieved accuracies are lower than those of logistic regression.

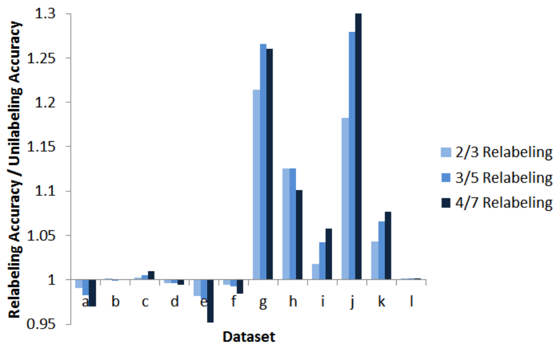

Figure 12: Unilabeling obtains better classifiers in some datasets, even when the workers are moderately accurate (p=0.75).

When comparing Figures 1 and 12 we observe that relabeling performance has improved with higher worker accuracy. This result is directly explained by the analysis in the section on worker accuracy. We see that 75% accuracy results in quick improvement of training quality using majority vote; 55% accuracy is a bad setting since relabeling only results in slow improvement.

Overall, we find that the experiments on real datasets confirm the trends discussed earlier and shown in our simulation experiments.

Related Work

A large body of work develops various methods for aggregating labels. For example, see (Dawid and Skene 1979; Whitehill et al. 2009; Raykar et al. 2010; Lin, Mausam, and Weld 2012a). Of note is BBMC (Wauthier and Jordan 2011), which develops a model that integrates active learning with data curation and model learning. Their algorithm can potentially trade off between relabeling and acquiring labels for new examples, but it is not general and is tied to their own classifier. Further, they do not consider this tradeoff. Researchers have also considered automated methods to decide when to relabel (Dai et al. 2013; Lin, Mausam, and Weld 2012a) but the goal is data accuracy instead of classifier accuracy.

Several researchers have considered *how* to pick examples or workers for (re)labeling when active learning or selective sampling (Donmez, Carbonell, and Schneider 2009; Donmez and Carbonell 2008; Donmez, Carbonell, and Schneider 2010; Yan et al. 2011; Dekel, Gentile, and Sridharan 2010; Sheng, Provost, and Ipeirotis 2008; Zhao, Sukthankar, and Sukthankar 2011). However, unlike our work, these do not answer the fundamental question of *when* to relabel.

Agnostic Learning (Kearns, Schapire, and Sellie 1994; Balcan, Beygelzimer, and Langford 2006; Golovin, Krause, and Ray 2010) is a general learning setting that makes little to no assumptions about learners and datasets. In this setting, noise refers to the labels that are inconsistent with the best hypothesis that is available to the learner. Thus agnostic learning can be viewed as a setting in which the goal is to train a classifier that fits both the noise and the data as well as possible. This scenario is inherently different than the one we consider, where noise is an incorrect label, not an inconsistent one, and we want to learn a classifier that fits the ground truth despite the noise.

Many works (*e.g.* (Natarajan, Dhillon, and Ravikumar 2013; Khardon and Wachman 2007; Crammer, Kulesza, and Dredze 2009)) design noise-tolerant classifiers. Indeed, evidence suggests that classifiers with convex loss functions, like logistic regression, are intolerant to label noise (Ding and Vishwanathan 2010). However, these works are orthogonal to ours in purpose. We focus on the tradeoff between unilabeling and relabeling for any black-box classifier. Our results can inform the relabeling strategy for noise-tolerant classifiers.

Several works seek to understand the sample complexity of classifiers under noise. (Laird 1988; Angluin and Laird 1988) derive bounds for classifiers that minimize their training error. The Statistical Query Model (Kearns 1993) can show that many PAC learning algorithms can be transformed into ones which tolerate classification noise.

Conclusion and Future Work

We have shown that when using crowdsourcing to learn the most accurate classifier possible with a fixed budget, relabeling examples should not be a default go-to strategy, as unilabeling often results in higher accuracies. We provide theoretical justification and empirical evidence using simulated and real datasets to show the following:

- Relabeling provides the most benefit to expressive classifiers with weak inductive bias. When the classifier being trained is linear, a relabeling strategy (and, indeed, higher levels of redundancy) is more likely to be appropriate when the domain has a large number of features.
- Typical relabeling strategies provide the most benefit when workers are moderately accurate, and not when they are extremely error-prone, as one might naively suspect. Unilabeling is preferred when workers are very accurate.
- As the labeling budget increases, unilabeling provides increasing benefits, but relabeling is often the more effective strategy when the budget is small.

Important for future work is a relaxation of our assumptions. We have assumed that all workers have the same accuracy. However, when workers have variable skill levels that are unknown to the controller, the utility of relabeling should increase, because the redundancy provided by relabeling allows aggregation strategies to jointly label data and calculate worker accuracies (Whitehill et al. 2009). Bad workers may simply be ignored. Determining the optimal relabeling policy, in this case, will require confronting an exploration / exploitation tradeoff.

Additionally, we have only considered passive learning approaches in our analysis of relabeling. However, results may drastically change when using more intelligent active learning approaches. Active learning could not only provide greater flexibility to pick interesting training examples, but also leverage the possibility that some examples may be worth more than others to denoise. Can we create an end to end decision-theoretic system that, given a new problem, will automatically query the appropriate examples for a new label or relabeling and output the best-quality classifier obtainable for a given labeling budget?

Acknowledgements

We thank Pedro Domingos, Carlos Guestrin, Luke Zettlemoyer, Eric Horvitz, Xiao Ling, and Jonathan Bragg for helpful discussions. We thank the anonymous reviewers for their comments. This work was supported by the WRF/TJ Cable Professorship, Office of Naval Research grant N00014-12-1-0211 and National Science Foundation grants IIS-1016713, IIS-1016465, and IIS-1420667.

References

Angluin, D., and Laird, P. 1988. Learning from noisy examples. *Machine Learning* 2(4):343–370.

Aslam, J. A., and Decatur, S. E. 1998. General bounds on statistical query learning and pac learning with noise via hypothesis boosting. *Information and Computation* 141(2):85–118.

Bache, K., and Lichman, M. 2013. UCI machine learning repository.

Balcan, M.-F.; Beygelzimer, A.; and Langford, J. 2006. Agnostic active learning. In *ICML*.

Crammer, K.; Kulesza, A.; and Dredze, M. 2009. Adaptive regularization of weight vectors. In *NIPS*.

Dai, P.; Lin, C. H.; Mausam; and Weld, D. S. 2013. Pomdp-based control of workflows for crowdsourcing. *Artificial Intelligence* 202:52–85.

Dawid, A., and Skene, A. M. 1979. Maximum likelihood estimation of observer error-rates using the em algorithm. *Applied Statistics* 28(1):20–28.

Dekel, O.; Gentile, C.; and Sridharan, K. 2010. Robust selective sampling from single and multiple teachers. In *COLT*.

Ding, N., and Vishwanathan, S. 2010. t-logistic regression. In *NIPS*.

Donmez, P., and Carbonell, J. G. 2008. Proactive learning: cost-sensitive active learning with multiple imperfect oracles. In *CIKM*, 619–628.

Donmez, P.; Carbonell, J. G.; and Schneider, J. 2009. Efficiently learning the accuracy of labeling sources for selective sampling. In *KDD*.

Donmez, P.; Carbonell, J. G.; and Schneider, J. 2010. A probabilistic framework to learn from multiple annotators with time-varying accuracy. In *SIAM International Conference on Data Mining (SDM)*, 826–837.

Dudley, R. M. 1978. Central limit theorems for empirical measures. *The Annals of Probability* 6(6):899–929.

Golovin, D.; Krause, A.; and Ray, D. 2010. Near-optimal bayesian active learning with noisy observations. In *NIPS*.

Ipeirotis, P. G.; Provost, F.; Sheng, V. S.; and Wang, J. 2013. Repeated labeling using multiple noisy labelers. *Data Mining and Knowledge Discovery* 28(2):402–441.

Jones, E.; Oliphant, T.; Peterson, P.; et al. 2001. SciPy: Open source scientific tools for Python.

Kearns, M. J.; Schapire, R. E.; and Sellie, L. M. 1994. Toward efficient agnostic learning. *Machine Learning* 17:115–141.

Kearns, M. J. 1993. Efficient noise-tolerant learning from statistical queries. In *ACM Symposium on the Theory of Computing*.

Khardon, R., and Wachman, G. 2007. Noise tolerant variants of the perceptron algorithm. *Journal of Machine learning Resesarch* 8:227–248.

Laird, P. D. 1988. *Learning from Good and Bad Data*. Kluwer International Series in Engineering and Computer Science. Kluwer Academic Publishers.

Lin, C. H.; Mausam; and Weld, D. S. 2012a. Crowdsourcing control: Moving beyond multiple choice. In *UAI*.

Lin, C. H.; Mausam; and Weld, D. S. 2012b. Dynamically switching between synergistic workflows for crowdsourcing. In *AAAI*.

Natarajan, N.; Dhillon, I. S.; and Ravikumar, P. 2013. Learning with noisy labels. In *NIPS*.

Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; Vanderplas, J.; Passos, A.; Cournapeau, D.; Brucher, M.; Perrot, M.; and Duchesnay, E. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12:2825–2830.

Raykar, V. C.; Yu, S.; Zhao, L. H.; and Valadez, G. 2010. Learning from crowds. *Journal of Machine Learning Research* 11:1297–1322.

Sheng, V. S.; Provost, F.; and Ipeirotis, P. G. 2008. Get another label? improving data quality and data mining using multiple, noisy labelers. In *Proceedings of the Fourteenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Snow, R.; O'Connor, B.; Jurafsky, D.; and Ng, A. 2008. Cheap and fast — but is it good? evaluating non-expert annotations for natural language tasks. In *EMNLP'08*.

Vapnik, V. N., and Chervonenkis, A. Y. 1971. On the uniform converge of relative frequencies of events to their probabilities. *Theory of Probability and its Applications* 16(2):264–280.

Wauthier, F. L., and Jordan, M. I. 2011. Bayesian bias mitigation for crowdsourcing. In *NIPS*.

Whitehill, J.; Ruvolo, P.; Bergsma, J.; Wu, T.; and Movellan, J. 2009. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In *NIPS*.

Yan, Y.; Rosales, R.; Fung, G.; and Dy, J. G. 2011. Active learning from crowds. In *ICML*.

Zhao, L.; Sukthankar, G.; and Sukthankar, R. 2011. Incremental relabeling for active learning with noisy crowdsourced annotations. In *IEEE Conference on Social Computing*.