Reactive Learning: Actively Trading Off Larger Noisier Training Sets Against Smaller Cleaner Ones

Christopher H. Lin CHRISLIN@CS.WASHINGTON.EDU
University of Washington, Seattle, WA

Mausam MAUSAM@CSE.IITD.AC.IN
Indian Institute of Technology, Delhi, India

Daniel S. Weld WELD@CS.WASHINGTON.EDU
University of Washington, Seattle, WA

Abstract

One of the most popular uses of crowdsourcing is to provide training data for supervised machine learning algorithms. Because of imperfect workers, requesters commonly ask multiple workers to redundantly label each example. When the goal is to train the best classifier at the lowest cost, active learning can intelligently pick new examples to label. However, active learning fails to address a fundamental tradeoff. Instead of always gathering new labels for new examples, we can also relabel, by gathering more labels for old, labeled examples. In this paper, we introduce the new problem of reactive learning, a generalization of active learning in which we seek to understand the difference in marginal value between decreasing the noise of a training set, via relabeling, and increasing the size and diversity of a noisier training set, via labeling new examples. We show how traditional active learning does not suffice for reactive learning, present new algorithms designed for this new problem, and empirically show that these algorithms can effectively make this tradeoff.

1. Introduction

Training data annotation is a common crowdsourcing application on many labor markets, such as Mechanical Turk, as well as on internal crowdsourcing platforms at companies like Microsoft and Google. Since human workers are imperfect, requesters commonly ask multiple workers to redundantly label each example, because multiple workers can simulate an expert worker (Snow et al., 2008). Multiple labels can be aggregated using simple majority vote or more complex approaches (e.g., (Dawid & Skene, 1979; Whitehill et al., 2009; Lin et al., 2012b; Bragg et al., 2013; Zhou et al., 2014)), and these de-noised aggregate labels can be used to train a learning algorithm. However, given a limited budget, "relabeling" data may not always be the best policy, if the goal is to train the best classifier possible.

(Lin et al., 2014) show in this setting that classifiers trained via "unilabeling," when m examples are each labeled only once, can often achieve higher accuracies compared to fixed relabeling strategies, when m/k examples are each labeled k times. In other words, they show that interestingly, having a larger, noisier training set can be more useful than a smaller, cleaner one.

However, they also show in many cases the opposite—that smaller, cleaner data sets train better classifiers than larger, noisier ones, thereby exposing a fundamental tradeoff. While they identify several characteristics of learning problems that can affect this tradeoff, they do not address control, leaving open what we call the *reactive learning* problem: given the current state of a learning algorithm, a fixed budget, and a pool of unlabeled *and labeled* examples, which example should one label *or relabel* next in order to maximize the classifier's accuracy? Although the vast literature on active learning under label noise addresses how to dynamically pick new examples to label, none of it considers the possibility of relabeling old examples. Reactive learning is a generalization of active learning that allows for the relabeling of examples.

We make the following contributions: 1) We show that standard active learning does not suffice for reactive learning, because it does not make full use of the available knowledge about the labels, and can easily become trapped

Proceedings of the 32^{nd} International Conference on Machine Learning, Lille, France, 2015. JMLR: W&CP volume 37.

by the assumption that a new example is always picked. 2) We introduce two classes of algorithms for reactive learning. One is a new measure of uncertainty for uncertainty sampling, and another, which we call *impact sampling*, tries to pick examples that will have the most impact on the classifier. These new algorithms are intuitive and easy to implement, and address the weaknesses of existing active learning algorithms. 3) We show in a number of empirical experiments that our new algorithms significantly outperform the state-of-the-art in active learning.

2. Related Work and Preliminaries

(Wauthier & Jordan, 2011) consider 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 custom classifier.

Several researchers have considered *how* to pick examples or workers for (re)labeling when active learning or selective sampling (Donmez et al., 2009; Donmez & Carbonell, 2008; Donmez et al., 2010; Yan et al., 2011; Dekel et al., 2010; Sheng et al., 2008; Zhao et al., 2011). However, unlike our work, these do not answer the fundamental question of *when* to relabel, or whether relabeling is even necessary. Researchers have also considered automated methods to decide when to relabel (Dai et al., 2013; Lin et al., 2012a; Bragg et al., 2014; Kamar & Horvitz, 2015) but the goal is data accuracy instead of classifier accuracy.

Agnostic Active Learning (Kearns et al., 1994; Balcan et al., 2006; Golovin et al., 2010) is a general learning setting in which noise can be adversarial, or be introduced by labelers, or refer to labels that are inconsistent with the best hypothesis that is available to the learner. However, like the rest of the active learning literature, they do not consider the possibility of relabeling examples. Many works (e.g., (Natarajan et al., 2013; Khardon & Wachman, 2007)) design noise-tolerant classifiers. 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.

We now set up the framework for reactive learning. Let $\mathcal{X}$ denote the space of examples, $\mathcal{Y} = \{0, 1\}$ a set of labels, and D, a distribution over $\mathcal{X}$. Let the true concept be $h^* : \mathcal{X} \to \mathcal{Y}$. Let $\mathcal{H}$ be a class of hypotheses, from which our learning algorithm, $\mathcal{C}$, tries to learn the $h \in \mathcal{H}$ that minimizes the error $\epsilon(h) = P_{x\sim D}(h(x) \neq h^*(x))$. Acquiring a label from a worker for an example incurs a fixed unit cost. We assume that each worker exhibits the same accuracy $a \in (0.5, 1]$, an assumption known as the *classification noise model* (Angluin & Laird, 1988) and that each label we acquire is flipped from the true label, $h^*(x)$, with probability $1-a$. We assume worker errors are independent and that the error rates are known.

Let $\mathcal{X}_L \subseteq \mathcal{X}$ denote the set of examples for which we currently have labels, $\mathcal{X}_U \subseteq \mathcal{X}$ denote the set of unlabeled examples, and for each $x_i \in \mathcal{X}$ let $l(x_i) = \{l_i^1, \ldots, l_i^{\tau_i}\}$ be the multiset of labels for that example, where τ_i is the number of labels we have for x_i. Let $f(l(x_i))$ output an aggregated label (e.g., majority vote) for an example given the noisy labels for that example. We train $\mathcal{C}$ using $\mathcal{X}_L$ and the corresponding aggregated labels output by f.

3. Algorithms For Reactive Learning

3.1. Uncertainty Sampling

Uncertainty sampling (Lewis & Catlett, 1994) is one of the most popular algorithms for active learning (Settles, 2012). To pick the next example to label, it simply computes a measure of the classifier's uncertainty (e.g., margin-based, entropy) for each example in the unlabeled set, $\mathcal{X}_U$, and then returns the most uncertain one. We denote as `US` uncertainty sampling that returns the $x \in \mathcal{X}_U$ with highest entropy.

We can directly apply uncertainty sampling to reactive learning by allowing it to sample from both $\mathcal{X}_U$ and $\mathcal{X}_L$. Let this algorithm be denoted `Re-US`. Unfortunately, `Re-US` can result in extremely poor performance. The problem is that in many cases, the most uncertain example (according to the classifier) will be a labeled example that we actually *are* certain about (according to the labels we have). For instance, suppose `Re-US` returns a point x_i close to the decision boundary of our classifier. But suppose further that this point has already been labeled a large number of times (τ_i is large). Relabeling this point is an extremely poor idea, since requesting another label will be highly unlikely to change the aggregated label $f(l(x_i))$, resulting in no change to the classifier, resulting in the same point being queried at the next time-step, forming an infinite loop during which no learning takes place.

Clearly, any reactive learning algorithm needs to consider both the classifier's uncertainty, which we now denote M_C, and the *label's* uncertainty, which we denote M_L. Thus, we propose a new uncertainty measure, which is a weighted average of these two uncertainties: $(1 - \alpha)M_C + \alpha M_L$, where $\alpha \in [0, 1]$. We define $M_L(x_i)$ as follows. For every example x_i, we compute label posteriors $P(h^*(x_i) \mid l(x_i))$ by applying Bayes' rule to the observed labels. Then, we use the entropy of these posteriors as the label's uncertainty: $M_L(x_i) = -\sum_{y\in\mathcal{Y}} P(h^*(x_i) = y \mid l(x_i)) \log P(h^*(x_i) = y \mid l(x_i))$. We denote this new algorithm `Re-US`(α). Unfortunately, learning the hyperparameter, α, may not be easy. Thus, we propose another

class of reactive learning algorithms, which do not require any hyperparameter learning.

3.2. Impact Sampling

Whereas our previous uncertainty sampling algorithm explicitly considers the knowledge contained in both the classifier and the labels, the class of impact sampling algorithms takes an indirect approach. Impact sampling algorithms simply pick the next example to (re)label that will impact the classifier the most, the intuition being that an example that heavily impacts the learned classifier must be a good example. Algorithm 1 describes the framework for computation of the impact of an example x. First, we train our classifier using all our labeled data, giving us a baseline hypothesis, h. Then, we train one classifier supposing that we received the label 0 for example x, giving us hypothesis h_0, and another classifier supposing that we received the label 1 for example x, giving us hypothesis h_1. We compare the predictions of h_1 and h_0 against the predictions of h on the labeled and unlabeled examples, and compute the number of predictions that changed for each, to give us $impact_0$, and $impact_1$. Finally, we return some function of $impact_0$ and $impact_1$. Different instantiations of impact sampling implement `retrain` and `weightedImpact` in various ways, which we now describe.

Algorithm 1 Impact Sampling

Input: Classifier $\mathcal{C}$, Example $x \in \mathcal{X}$, Unlabeled Examples $\mathcal{X}_U$, Labeled Examples $\mathcal{X}_L$ and their corresponding aggregated labels as given by f and l.
Initialize $impact_0 = 0$, $impact_1 = 0$.
$h = \texttt{retrain}(\mathcal{C}, \mathcal{X}_L, f, l, \_, \_)$
$h_1 = \texttt{retrain}(\mathcal{C}, \mathcal{X}_L, f, l, x, 1)$.
$h_0 = \texttt{retrain}(\mathcal{C}, \mathcal{X}_L, f, l, x, 0)$.
for $x_i \in \mathcal{X}_\mathcal{U} \cup \mathcal{X}_\mathcal{L}$ **do**
  if $h_0(x_i) \neq h(x_i)$ **then**
    $impact_0 = impact_0 + 1$
  end if
  if $h_1(x_i) \neq h(x_i)$ **then**
    $impact_1 = impact_1 + 1$
  end if
end for
Return $\texttt{weightedImpact}(impact_0, impact_1)$

3.2.1. Optimism

The most straightforward way to implement `weightedImpact` is to use the label posteriors (computed using the classifier's beliefs as a prior) to compute an expectation of $impact_0$ and $impact_1$: $\sum_{y \in \mathcal{Y}} a \cdot P(h^*(x_i) = y \mid l(x_i)) \cdot impact_y$. We denote these impact sampling algorithms with `EXP`. However, when training a classifier with noisy labels, the learned classifier may output beliefs that cannot be trusted. In these cases, injecting impact sampling with some optimism can be helpful. We can implement `weightedImpact` so that instead of returning an expected impact, it returns the maximum impact: $\max(impact_0, impact_1)$. Taking a maximum makes impact sampling optimistic in that now it assumes the largest possible impact for any given example. We denote impact sampling with optimism with `OPT`.

3.2.2. Pseudo-Lookahead

The most straightforward way to implement `retrain` is to simply add the new fake labels that we pretend to have received for x_i into the multiset $l(x_i)$. Thus, the algorithm is myopic in that for certain multisets, adding this additional label may have no effect on the aggregated label $f(l(x_i))$ at all, and consequently no effect on the learned classifier. For example, if we are using majority vote as f and we currently have 3 votes in favor of the label 1 and 1 vote in favor of the label 0, one additional vote for 0 will result in an $impact_0$ of 0. Myopicity is problematic because training the classifier optimally may require gathering multiple labels for the same example. To alleviate this problem, we can introduce the ability to perform a "pseudo-lookahead."

Whenever we are considering an example $x_i \in \mathcal{X}_L$ from the labeled set (we never have the myopicity problem when we are considering a new unlabeled example), we implement the `retrain` function so that instead of just adding the new label l_i^{new} into the current multiset $l(x_i)$, we ensure that the classifier is trained with l_i^{new} as the aggregated label, instead of $f(l(x_i))$. Then, we implement `weightedImpact` so that we divide the computed impact of that label, $impact_{l_i^{new}}$, by the minimum of 1 and the smallest number of additional worker labels that would have been needed to flip the aggregated label $f(l(x_i))$. Intuitively, we are effectively computing a normalized impact of another label l_i^{new}, given we train $\mathcal{C}$ with l_i^{new}. We denote algorithms that use pseudo-lookahead with `PL`.

4. Experiments

We now present empirical experiments to compare the performances of various impact sampling algorithms, `Re-US(`α`)`, and `US`. We aim to answer three questions: what setting of α is best, what kind of impact sampling is best, and whether our methods are more effective at reactive learning than `US`. We also look at the performance of `US-perfect`, which runs `US` with perfect data, providing a benchmark on achievable accuracy.

To reduce computational costs, we only allow impact sampling to choose among two points instead of $\mathcal{X}$: the point recommended by `US`, and the point recommended

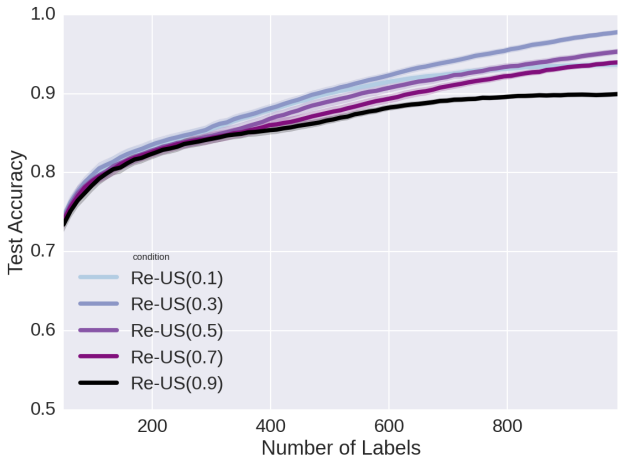

Figure 1. Generalization accuracy of logistic regression when trained using Re-US(α) with $\alpha \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$.

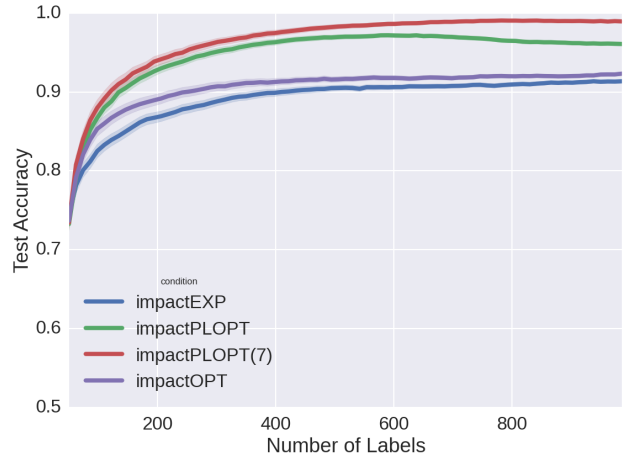

Figure 2. Generalization accuracy of logistic regression when trained using various impact sampling strategies.

by `US` applied to only $\mathcal{X}_L$. We also try versions that can choose among seven points: the two points as before, and the five points returned by `Re-US(`α`)` where $\alpha \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$.

We train an l2-regularized logistic regression. We use a synthetic domain that contains two random Gaussian clusters, which correspond to two classes. We generate a dataset by randomly picking two means, $\mu_1, \mu_2 \in [0, 1]^k$, and two corresponding covariance matrices $\Sigma_1, \Sigma_2 \in [0, 1]^{k \times k}$. For each Gaussian cluster (class), we generate 1,000 examples. All experiments are averaged over 1,000 random datasets. We vary the number of features among $\{10, 30, 50, 70, 90\}$, but only show results for 90 features, due to lack of space. We seed training with 50 examples, use a total budget of 1,000, and test on 300 examples. We set worker accuracy to be $a = 0.75$.

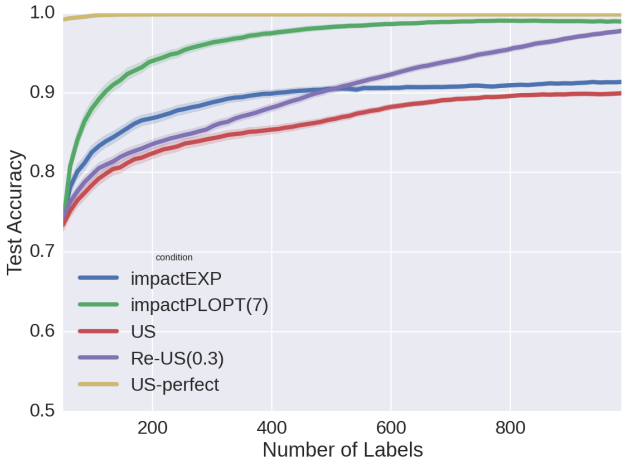

Figure 3. Generalization accuracy of logistic regression when trained using various impact sampling and uncertainty sampling strategies.

Figure 1 compares `Re-US(`α`)` against itself for $\alpha \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$. We see that `Re-US(0.3)` does well, and that as we increase α from 0.3, the performance of the learned classifier drops. Different α work best when we vary the number of features.

Figure 2 shows the effect of introducing optimism and psuedo-lookahead into impact sampling. We see that these techniques are able to substantially increase performance. First, when we consider impact sampling without pseudo-lookahead, we see that it works better with optimism (`impactOPT`) than without (`impactEXP`). Next, we see that adding pseudo-lookahead (`impactPLOPT`) is better than not (`impactOPT`). And finally, we see that allowing impact sampling to choose from a larger pool of candidate examples (`impactPLOPT(7)`) is quite beneficial.

Figure 3 compares impact sampling to uncertainty sampling. We see that even `impactEXP`, the weakest impact sampling strategy, strictly dominates `US`, the most popular active learning method, in the setting of reactive learning. We also see that `impactPLOPT(7)` is quite close to the accuracy of `US-perfect`, unlike the other methods.

We repeated these experiments using another synthetic dataset with three Gaussian clusters instead of two, and found extremely similar results.

5. Conclusion

We have presented reactive learning, a generalization of active learning that allows for the relabeling of examples. We have shown that standard active learning is not sufficient in this new setting, presented a new uncertainty measure and a new class of algorithms, impact sampling, and empirically shown their effectiveness for reactive learning.

References

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

Balcan, Maria-Florina, Beygelzimer, Alina, and Langford, John. Agnostic active learning. In *ICML*, 2006.

Bragg, Jonathan, Mausam, and Weld, Daniel S. Crowdsourcing multi-label classification for taxonomy creation. In *HCOMP*, 2013.

Bragg, Jonathan, Mausam, and Weld, Daniel S. Parallel task routing for crowdsourcing. In *HCOMP*, 2014.

Dai, Peng, Lin, Christopher H., Mausam, and Weld, Daniel S. Pomdp-based control of workflows for crowdsourcing. *Artificial Intelligence*, 202:52–85, 2013.

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

Dekel, Ofer, Gentile, Claudio, and Sridharan, Karthik. Robust selective sampling from single and multiple teachers. In *COLT*, 2010.

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

Donmez, Pinar, Carbonell, Jaime G., and Schneider, Jeff. Efficiently learning the accuracy of labeling sources for selective sampling. In *KDD*, 2009.

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

Golovin, Daniel, Krause, Andreas, and Ray, Debajyoti. Near-optimal bayesian active learning with noisy observations. In *NIPS*, 2010.

Kamar, Ece and Horvitz, Eric. Planning for crowdsourcing hierarchical tasks. In *AAMAS*, 2015.

Kearns, Michael J., Schapire, Robert E., and Sellie, Linda M. Toward efficient agnostic learning. *Machine Learning*, 17:115–141, 1994.

Khardon, Roni and Wachman, Gabriel. Noise tolerant variants of the perceptron algorithm. *Journal of Machine Learning Research*, 8:227–248, 2007.

Lewis, David D. and Catlett, Jason. Heterogeneous uncertainty sampling for supervised learning. In *ICML*, 1994.

Lin, Christopher H., Mausam, and Weld, Daniel S. Dynamically switching between synergistic workflows for crowdsourcing. In *AAAI*, 2012a.

Lin, Christopher H., Mausam, and Weld, Daniel S. Crowdsourcing control: Moving beyond multiple choice. In *UAI*, 2012b.

Lin, Christopher H., Mausam, and Weld, Daniel S. To re(label), or not to re(label). In *HCOMP*, 2014.

Natarajan, Nagarajan, Dhillon, Inderjit S., and Ravikumar, Pradeep. Learning with noisy labels. In *NIPS*, 2013.

Settles, Burr. *Active Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2012.

Sheng, Victor S., Provost, Foster, and Ipeirotis, Panagiotis G. 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*, 2008.

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

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

Whitehill, Jacob, Ruvolo, Paul, Bergsma, Jacob, Wu, Tingfan, and Movellan, Javier. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In *NIPS*, 2009.

Yan, Yan, Rosales, Romer, Fung, Glenn, and Dy, Jennifer G. Active learning from crowds. In *ICML*, 2011.

Zhao, Liyue, Sukthankar, Gita, and Sukthankar, Rahul. Incremental relabeling for active learning with noisy crowdsourced annotations. In *IEEE Conference on Social Computing*, 2011.

Zhou, Dengyong, Liu, Qiang, Platt, John C., and Meek, Christopher. Aggregating ordinal labels from crowds by minimax conditional entropy. In *ICML*, 2014.