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

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. 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.

Preliminaries Let $\mathcal{X}$ denote the space of examples, $\mathcal{Y} = \{0, 1\}$ a set of labels, $\mathcal{X}_L \subseteq \mathcal{X}$ 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 using $\mathcal{X}_L$ and the corresponding aggregated labels output by f. Acquiring a label incurs a fixed unit cost. We assume that each worker exhibits the same accuracy $a \in (0.5, 1]$, and that worker errors are independent and that a is known. Given the current l, the goal of *reactive learning* is to select an example $x \in \mathcal{X}$ (not $x \in \mathcal{X}_U$, as in traditional active learning) such that acquiring a label for x and adding it to l minimizes the long-term error of the trained classifier.

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).

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. Different instantiations of impact sampling implement `retrain` and `weightedImpact` in various ways, which we now describe.

Optimism The most straightforward way to implement `weightedImpact` is to use the label posteriors (computed

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 $\texttt{weightedImpact}(impact_0, impact_1)$

using the classifier's beliefs as a prior) to compute an expectation of $impact_0$ and $impact_1$: $\sum_{y \in \mathcal{Y}}[aP(h^*(x_i) = y \mid l(x_i)) + (1-a)P(h^*(x_i) \neq y \mid l(x_i))] \cdot impact_y(x_i)$. 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`.

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`.

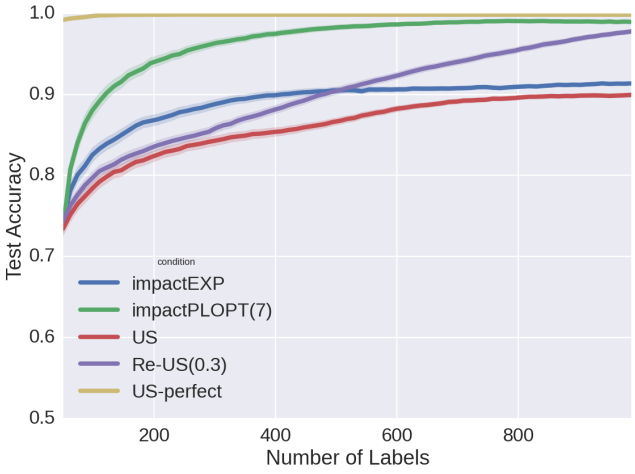

Figure 1: Generalization accuracy of logistic regression when trained using various impact sampling and uncertainty sampling strategies.

Experiments 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 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 set $k = 90$ features. We seed training with 50 examples, use a total budget of 1,000, and test on 300 examples. We set worker accuracy to 0.75%.

Figure 1 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`, which runs `US` with perfect data, providing a benchmark on achievable accuracy. We repeated these experiments using another synthetic dataset with three Gaussian clusters, and found extremely similar results.

References

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

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