# Re-active Learning: Active Learning with Relabeling

**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

Active learning seeks to train the best classifier at the lowest annotation cost by intelligently picking the best examples to label. Traditional algorithms assume there is a single annotator and disregard the possibility of requesting additional independent annotations for a previously labeled example. However, relabeling examples is important, because all annotators make mistakes — especially crowdsourced workers, who have become a common source of training data. This paper seeks to understand the difference in marginal value between decreasing the noise of the training set via relabeling and increasing the size and diversity of the (noisier) training set by labeling new examples. We use the term *re-active learning* to denote this generalization of active learning. We show how traditional active learning methods perform poorly at re-active learning, present new algorithms designed for this important problem, formally characterize their behavior, and empirically show that our methods effectively make this tradeoff.

## Introduction

In order to minimize the expense of labeling training data, active learning algorithms reason about the best examples to annotate. Traditional active-learning methods assume a *single* annotator, either perfect (Settles 2012) or noisy (Kearns, Schapire, and Sellie 1994). In this setting, such algorithms always pick a *new* example to label, since they can gather no new information about examples which have already been labeled. However, the single annotator assumption is increasingly violated because machine learning practitioners routinely employ multiple annotators (crowdsourced or professional) to label each training example in order to increase its accuracy (e.g., (Snow et al. 2008; LDC 2015)). In response, modern active learning algorithms must deal with a fundamentally new tradeoff: whether to gather the first label for an unlabeled example, or instead reduce the noise of an existing labeled example by asking for an additional annotation of that example.

Previous work identifies this tradeoff (Sheng, Provost, and Ipeirotis 2008; Ipeirotis et al. 2013), but does not deliver a solution, instead assuming no unlabeled examples and restricting attention to the choice of the best example to relabel. Another method for learning with multiple, noisy annotators *never* relabels examples, and instead, finds the best worker for a new example so that it is likely labeled at a high accuracy (Yan et al. 2011). Yet another approach suggests a single level of redundancy for each label, and samples points uniformly (Lin, Mausam, and Weld 2014). To our knowledge, no existing method answers the important question: "Which data point should one actively label or relabel in order to learn the best classifier?" We call this generalization of active learning "re-active learning."

Standard active learning strategies like uncertainty sampling (Lewis and Catlett 1994) and expected error reduction (Roy and McCallum 2001; Kapoor, Horvitz, and Basu 2007) can be naïvely extended for re-active learning by allowing them to pick *any* (labeled or unlabeled) point using their existing approaches. Unfortunately, we show that such extensions do not perform well, because they do not utilize all sources of knowledge, are myopic, or both. These extensions often suffer from infinite looping, when the learner repeatedly annotates the same example, starving others. In addition to considering information gleaned from the classifier, a re-active learner must also leverage information from the number and agreement of the gathered labels themselves.

Therefore, we introduce alternative extensions of uncertainty sampling that can reason about the information in the labels. We also propose a new class of algorithms, *impact sampling*, which picks the example to label that has the potential to change the classifier the most. By reasoning about whether an additional label can change the classifier, impact sampling elegantly eliminates the starvation problems described above. Many active learning methods use greedy search to reduce combinatorial explosion, but the resulting myopia is especially problematic with relabeling — if the first two labels agree, then a third may have no effect. To combat this problem, we introduce a novel technique, called pseudo-lookahead, that can tractably mitigate myopia. We then characterize the relationship between impact sampling and uncertainty sampling, and show that, surprisingly, in many noiseless settings, impact sampling can be viewed as a generalization of uncertainty sampling. Finally, we conduct empirical experiments on both synthetic and real-world datasets, showing that our new algorithms significantly outperform traditional active learning techniques and other natural baselines on the problem of re-active learning.

Copyright © 2016, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

## Previous Work

Prior work (Sheng, Provost, and Ipeirotis 2008; Ipeirotis et al. 2013) identifies the tradeoff between relabeling examples and gathering new examples. They show that relabeling examples is useful for supervised learning, and they also consider how to pick examples to relabel when relabeling is the only option. However, crucially, unlike our work, they do not consider any combination of relabeling and the labeling of new examples. Lin et al. (2014) investigate when and why relabeling is useful. In contrast to Sheng et al., they find that relabeling is not always useful. They find that learning problems with higher VC dimension, moderate levels of noise, and lower fixed budgets may benefit more from strategies with more relabeling.

Kääriäinen (2006) shows that if each label is independently corrupted by noise, then active learning with noise is theoretically almost as easy as active learning without noise, because one can repeatedly label each point a sufficient number of times to cancel out the noise. In contrast, we show that the small theoretical difference can be quite large in practice, and thus reasoning about whether or not to relabel is extremely important.

The active learning algorithm of Wauthier et al. (2011) can potentially trade off between relabeling and acquiring labels for new examples. However, they do not identify the tradeoff explicitly, nor explore it analytically or experimentally. Furthermore, their solution requires gold labels, and is tied to their own custom classifier.

Several researchers consider how to pick workers for either labeling (Donmez and Carbonell 2008; Yan et al. 2011) or relabeling (Dekel, Gentile, and Sridharan 2010). Unlike our work, none of these works answers the fundamental question of *when* to relabel, or whether or not relabeling is necessary. Researchers have also considered how to decide when to relabel when the objective is data accuracy (Dai et al. 2013; Lin, Mausam, and Weld 2012b; 2012a; Bragg, Mausam, and Weld 2014; Kamar and Horvitz 2015) or ranking (Chen et al. 2013) instead of the accuracy of a classifier learned using the data.

Agnostic Active Learning (Kearns, Schapire, and Sellie 1994; Balcan, Beygelzimer, and Langford 2006; Golovin, Krause, and Ray 2010) is a general learning setting for active learning with noise. However, like the rest of the active learning literature, it does not consider relabeling.

Zhang et al. (2015) consider an active learning setting where instead of relabeling, one can choose between querying an expert labeler who is more expensive or a noisy labeler who is cheaper. They learn a classifier that predicts when these labelers differ in their labels and only ask the expert when they predict disagreement.

On-the-job Learning (Werling et al. 2015) is an extension of online active learning that allows the learning algorithm to query the crowd (if doing so is likely to help) prior to making a prediction. In contrast, we focus on how to train the best classifier possible offline, and thus relabeling examples may not always be necessary.

Impact sampling is similar to optimized probabilistic active learning (Krempl, Kottke, and Lemaire 2015), which computes how label distributions locally change from querying additional examples.

Finally, impact sampling can be considered a generalization of selection by expected gradient length (Settles, Craven, and Ray 2007), which queries the example that would create the greatest change in the gradient of the objective function, but as we will show, impact sampling does not require knowledge of the classifier's objective function.

## Preliminaries

We now set up the framework for re-active 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, $\mathcal{A}$, tries to select the $h \in \mathcal{H}$ that minimizes the error $\epsilon(h) = P_{x \sim D}(h(x) \neq h^*(x))$. Following common practice for crowdsourcing labels, we assume that acquiring a label for an example incurs a fixed unit cost.

Let $\mathcal{X}_L \subseteq \mathcal{X}$ denote the current set of labeled examples, and $\mathcal{X}_U = \mathcal{X} - \mathcal{X}_L$ denote the set of unlabeled examples. Let $L = \{(x_i, y_i)\}$ denote the multiset of example and label pairs, and for each $x_i \in \mathcal{X}_L$, let $L_{x_i} = \{l_i^1, \ldots, l_i^{\tau_i}\}$ be the multiset of labels for $x_i$, where $\tau_i$ is the number of labels for $x_i$. Let $f(L_{x_i})$ output an aggregated label for an example given the noisy labels for that example. $f$ can be as simple as majority vote or use more complex statistical techniques. (e.g., (Dawid and Skene 1979; Whitehill et al. 2009; Lin, Mausam, and Weld 2012a)). We run our learning algorithm $\mathcal{A}$ using $L$ and the corresponding aggregated labels output by $f$. Given the current $L$, the goal of *re-active 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 classifier output by $\mathcal{A}$.

## Algorithms For Re-Active Learning

### Uncertainty Sampling

Uncertainty sampling (Lewis and 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 for each example in the unlabeled set, $\mathcal{X}_U$, and then returns the most uncertain one. Let $M_A(x_i)$ denote the entropy of the probabilities output by the learning algorithm after training on $L$, for an example $x_i$. We denote as $\texttt{US}_{\mathcal{X}_U}$ the strategy that returns the example in $\mathcal{X}_U$ with highest entropy: $\text{argmax}_{x \in \mathcal{X}_U} M_A(x)$.

One could naïvely apply uncertainty sampling to re-active learning by allowing it to sample from the full sample space $\mathcal{X} = \mathcal{X}_U \cup \mathcal{X}_L$. We denote this algorithm $\texttt{US}_{\mathcal{X}}$. Unfortunately, $\texttt{US}_{\mathcal{X}}$ can result in extremely poor performance.

Consider Figure 1. Suppose that $x_1$ and $x_2$ are the only labeled points in this distribution of diamonds and circles and $h$ is the current hypothesis. Then, $\texttt{US}_{\mathcal{X}}$ will pick $x_1$ or $x_2$ to relabel, because they are the closest to $h$, and thus have the highest uncertainty. In fact, $\texttt{US}_{\mathcal{X}}$ will likely pick $x_1$ or $x_2$ to relabel repeatedly, because each time it picks $x_1$ or $x_2$, it will receive a label that will most likely not change the aggregated label, $f(L_{x_i})$ (assuming low label noise). Since the

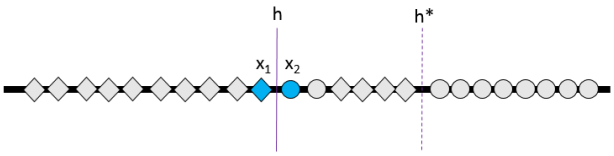


Figure 1: An example domain in which a naïve extension of uncertainty sampling to re-active learning, $\texttt{US}_{\mathcal{X}}$, can fall into a trap. Given that $x_1$ and $x_2$ are the only labeled points in this distribution of diamonds and circles and $h$ is the current hypothesis, uncertainty sampling is likely to converge to behavior in which it will always pick these same examples to relabel and cause an infinite loop in which no learning takes place. The true hypothesis, $h^*$, will never be learned.

labels used to train the classifier will not change, the classifier itself will not change, forming an infinite loop during which other useful points get starved and no learning takes place, causing convergence to a suboptimal hypothesis.

This weakness is similar to the hasty generalization problem found in active learning (Dasgupta and Hsu 2008), but is distinct, because points are not relabeled in traditional active learning. We find experimentally that $\texttt{US}_{\mathcal{X}}$ gets stuck in unproductive infinite loops quite often. The problem is that in many cases, the most uncertain example (according to the classifier) could be a labeled example, which is actually quite certain (according to the current label multiset).

**Extensions of Uncertainty Sampling for Re-Active Learning** Clearly, any extension of uncertainty sampling to re-active learning needs to consider both $M_A(x_i)$, the classifier's uncertainty, and the *label's* uncertainty, which we denote $M_L(x_i)$. We define $M_L(x_i)$ as the entropy of the labels themselves: $M_L(x_i) = -\sum_{y \in \mathcal{Y}} P(h^*(x_i) = y \mid L_{x_i}) \log P(h^*(x_i) = y \mid L_{x_i})$, where the label posterior $P(h^*(x_i) \mid L_{x_i})$ is computed by applying Bayes' rule to the observed labels $L_{x_i}$ on top of a uniform prior.

We propose a new aggregate uncertainty measure, which is a weighted average of these two uncertainties: $(1 - \alpha)M_A(x_i) + \alpha M_L(x_i)$, where $\alpha \in [0, 1]$. We denote this new algorithm, which picks the example $x_i$ with the highest aggregate uncertainty, as $\texttt{US}^{\alpha}_{\mathcal{X}}$. By definition, $\texttt{US}^{0}_{\mathcal{X}}$ is equivalent to $\texttt{US}_{\mathcal{X}}$ . We also note the following interesting and counter-intuitive fact that weighting *label* uncertainty highly (large $\alpha$) results in behavior equivalent to that of standard uncertainty sampling for active learning ($\texttt{US}_{\mathcal{X}_U}$), which only considers classifier uncertainty. The proof is in the Supplementary Materials.

**Theorem 1.** *For any given learning problem, there exists an* $\alpha' < 1.0$ *such that for all* $\alpha \in [\alpha', 1.0]$, $\texttt{US}^{\alpha}_{\mathcal{X}}$ *is equivalent to* $\texttt{US}_{\mathcal{X}_U}$.

An alternative way we can use the popular framework of uncertainty sampling to fit the new framework of re-active learning is to simply use $\texttt{US}_{\mathcal{X}_U}$, but label each new example a fixed number of times. We use $\texttt{US}^{j/k}_{\mathcal{X}_U}$ to denote the algorithm that picks a new example to label via $\texttt{US}_{\mathcal{X}_U}$, and then relabels that example using $j/k$-relabeling, where the algorithm can request up to $k$ labels, stopping as soon as $j = \lceil k/2 \rceil$ identical labels are received. Unfortunately, for both $\texttt{US}^{\alpha}_{\mathcal{X}}$ and $\texttt{US}^{j/k}_{\mathcal{X}_U}$, learning optimal values for hyperparameters, $\alpha$ or $k$, can be difficult.

## Expected Error Reduction

Expected error reduction (EER) (Roy and McCallum 2001; Kapoor, Horvitz, and Basu 2007) is another active learning technique that can be extended to the framework of re-active learning. EER simply chooses to label the example that maximizes the expected reduction in classifier error, where the expected error of a classifier is estimated using the probabilities output by that classifier.

To extend EER to consider relabeling, we must first compute the probability, $Q(y|x_i)$, that we receive label $y$ if we query example $x_i$. Let $P_{\mathcal{A}}(h^*(x_i) = y \mid L_{x_i})$ be the probability that the correct label of $x_i$ is $y$; we may compute it by first using the currently learned classifier's beliefs as a prior and then performing a Bayesian update with the observed labels $L_{x_i}$. Then, let $a$ denote label accuracy, the probability that the queried label will be correct. Then, $Q(y|x_i) = [aP_{\mathcal{A}}(h^*(x_i) = y \mid L_{x_i}) + (1 - a)P_{\mathcal{A}}(h^*(x_i) \neq y \mid L_{x_i})]$.

Now, let $L \oplus \{(x_i, y)\}$ denote the addition of the element $(x_i, y)$ to the multiset $L$, and let $\epsilon(\mathcal{A}|L)$ denote the error of the classifier returned by running $\mathcal{A}$ using the labels, $L$. Conceptually, EER returns $\operatorname{argmin}_{x \in \mathcal{X}}[\sum_{y \in \mathcal{Y}} Q(y|x)\epsilon(\mathcal{A}|L \oplus \{(x, y)\})] - \epsilon(\mathcal{A}|L)$, and it approximates $\epsilon$ using the probabilities output by the learned classifier.

Unlike $\texttt{US}_{\mathcal{X}}$, EER is able to incorporate knowledge about label uncertainty. Unfortunately, like $\texttt{US}_{\mathcal{X}}$, EER can starve points. Suppose that the expected reduction in error from querying any unlabeled point $x_u$ is negative. Such a result can be a common occurrence due to the myopic nature of EER and the amount of uncertainty in the problem. Next, suppose that the reduction in error from querying any labeled point $x_l$ is 0. Such a result is also common, since oftentimes a single extra label will not change the resulting aggregated labels output by $f$, especially if $x_l$ has already been labeled many times. In this scenario, EER will query $x_l$ over and over, starving valuable data points as did $\texttt{US}_{\mathcal{X}}$. The key problem is myopia, since the best next example to label may temporarily reduce the accuracy of the classifier.

This problem is exacerbated by the fact that full EER is computationally intractable. Just computing the expected future error for a single new label requires retraining a classifier twice. In order to make EER practical, one must restrict the set of examples from which EER may recommend points. However, reducing the number of points that EER may query from only increases the likelihood that EER falls prey to infinite looping.

## Impact Sampling

We now introduce a new class of algorithms, impact sampling, which addresses the problems with uncertainty sampling and EER. Impact sampling algorithms 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 to label. Let $h_L$ be

the hypothesis learned using learning algorithm $\mathcal{A}$, labels $L$, and aggregation function $f$. We define the *impact* of example $x_i$ given label $y$, $\psi_y(x_i)$, as the probability that adding the additional label $y$ for $x_i$ changes the predicted label of an example in $\mathcal{X}$: $\psi_y(x_i) = P_{x\sim\mathcal{D}}(h_L(x) \neq h_{L\oplus\{(x_i,y)\}}(x))$.

Algorithm 1 describes the framework for computing the impact of an example $x_i$. First, it trains the classifier using the labeled data, producing a baseline hypothesis, $h_L$. Then, it trains one classifier supposing that it received a label 0 for example $x_i$, producing hypothesis $h_{L0}$, and another classifier supposing that it received the label 1 for example $x_i$, producing hypothesis $h_{L1}$. Next, it computes $\psi_0(x_i)$ and $\psi_1(x_i)$ by taking a sample from $\mathcal{X}$ and comparing the predictions of $h_{L1}$ and $h_{L0}$ against the predictions of $h$ on that sample, and computing the fraction of predictions that changed for each. Alternatively, it can reason about how classifiers change with training. Finally, it returns some `weightedImpact`$(\psi_0(x_i), \psi_1(x_i))$, as the total impact of an example, $\psi(x_i)$. We construct variations of impact sampling by redefining $\oplus$ or implementing `weightedImpact` in various ways, which we now describe.

**Algorithm 1** Computation of Impact of an Example $x_i$

**Input:** Learning algorithm $\mathcal{A}$, Example $x_i \in \mathcal{X}$, Examples $\mathcal{X}$, aggregation function $f$, and label multiset $L$.
**Output:** Total impact of example $x_i$, $\psi(x_i)$
Initialize $\psi_0 = 0, \psi_1 = 0$.
$L1 = L \oplus \{(x_i, 1)\}$, $L0 = L \oplus \{(x_i, 0)\}$
$h_L = \texttt{retrain}(\mathcal{A}, f, L)$
$h_{L1} = \texttt{retrain}(\mathcal{A}, f, L1)$
$h_{L0} = \texttt{retrain}(\mathcal{A}, f, L0)$
**for** $x_j \in sample(\mathcal{X})$ **do**
  **if** $h_{L0}(x_j) \neq h_L(x_j)$ **then**
    $\psi_0 = \psi_0 + \frac{1}{|\mathcal{X}|}$
  **end if**
  **if** $h_{L1}(x_j) \neq h_L(x_j)$ **then**
    $\psi_1 = \psi_1 + \frac{1}{|\mathcal{X}|}$
  **end if**
**end for**
$\psi = \texttt{weightedImpact}(\psi_0, \psi_1)$
**Return** $\psi$

**Optimism** The most straightforward way to implement `weightedImpact` is to use label posteriors to compute the total expected impact $\psi(x_i) = \sum_{y\in\mathcal{Y}} Q(y|x_i) \cdot \psi_y(x_i)$. We use `EXP` to denote impact sampling algorithms that compute an expected impact (e.g., `impactEXP`).

However, when training a classifier with noisy labels, the learned classifier often outputs beliefs that are wildly wrong. This can mislead the expected impact calculations and starve certain examples that could substantially improve the classifier, all because the classifier believes the impactful labels are unlikely. Furthermore, for most labeled examples in $\mathcal{X}_L$, at least one of $\psi_0$ and $\psi_1$ will be 0, biasing `impactEXP` to undervalue their impact over that of an unlabeled point. Computing the expectation also requires knowledge of label accuracy, which, like in EER, must be learned. To mitigate these we inject impact sampling with some optimism by implementing `weightedImpact` so that instead of returning the expected impact, it instead returns the *maximum impact*: $\psi = \max(\psi_0, \psi_1)$. This leads the system to pick the example that could produce the largest possible impact on the classifier. We use `OPT` to denote impact sampling with optimism.

**Pseudo-Lookahead** The most straightforward way to implement $\oplus$ is to simply add the new hypothetical labels for $x_i$ into the multiset $L_{x_i}$. However, such an implementation makes the algorithm myopic, because for certain multisets, a single additional label may have no effect on the aggregated label, $f(L_{x_i})$, and thus no effect on the learned classifier. For example, if $f$ is majority vote and $L_{x_i}$ currently has 2 positive votes and 0 negative votes, any single new annotation will have zero impact. Myopicity is problematic because correcting labeling mistakes may require gathering multiple labels for the same example. To alleviate this problem, we introduce the "pseudo-lookahead."

Whenever impact sampling is considering an example $x_i \in \mathcal{X}_L$ from the labeled set (the myopicity problem does not exist when considering a new unlabeled example), we implement the $\oplus$ operator so that instead of directly 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 $f(L_{x_i})$. Let $\rho$ be the minimum number of additional labels needed to flip the aggregate label to $l_i^{new}$ ($\rho$ is zero if $l_i^{new}$ is already $f(L_{x_i})$). We implement `weightedImpact` so that the computed impact of that label, $\psi_{l_i^{new}}$, is divided by $\max(1, \rho)$ before any additional computation: $\psi_{l_i^{new}} = \psi_{l_i^{new}} / \max(1, \rho)$. Intuitively, this pseudo-lookahead is computing a normalized impact of a single label, if impact sampling trains $\mathcal{A}$ with aggregate label $l_i^{new}$ after receiving multiple such labels. We denote algorithms that use pseudo-lookahead with `PL`. We note that introducing pseudo-lookahead can theoretically cause impact sampling to starve certain examples of labels, but that we almost never see this behavior in our experimentation.

**Implementation Issues** We can construct four different impact sampling algorithms with these different implementations of $\oplus$ and `weightedImpact`: `impactEXP`, `impactOPT`, `impactPLEXP` and `impactPLOPT`. In practice, optimism and pseudo-lookahead bias towards more relabeling, so we implement them for computing impacts of labeled examples only.

Computing the impact of a single point can require retraining a classifier twice (much like EER), and thus picking a single point to (re)label can require up to $2|\mathcal{X}|$ retrainings. To speedup the implementation, we restrict impact sampling to choose only between 2 points instead of all of $\mathcal{X}$: the point recommended by $\texttt{US}_{\mathcal{X}_U}$, and the point recommended by $\texttt{US}_{\mathcal{X}_L}$ (uncertainty sampling applied to only $\mathcal{X}_L$.) We also try versions that choose among 7 points: the 2 points as before, and the 5 points returned by $\texttt{US}^{\alpha}_{\mathcal{X}}$ where $\alpha \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$. In the experiments, we indicate the number of points from which the strategy draws in parentheses.

Computing the impact also requires iteration over a sample of $\mathcal{X}$. Significant speedups may be achieved by sampling examples from regions that that are likely to have been af-

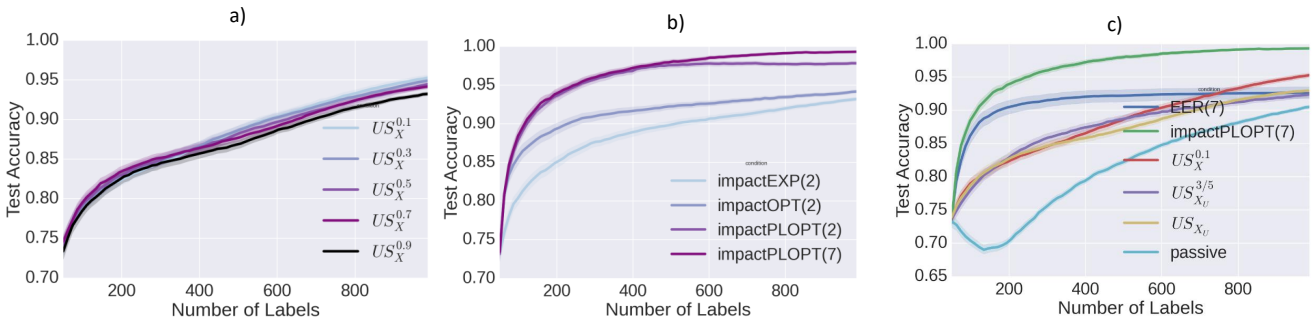


Figure 2: Average generalization accuracy of logistic regression over randomly generated Gaussian datasets with 90 features and label accuracy 0.75, when trained using various strategies. a) compares various settings of $\texttt{US}_{\mathcal{X}}^{\alpha}$ and shows $\alpha = 0.1$ is best. b) compares various impact sampling strategies and shows `impactPLOPT(7)` is best. c) compares impact sampling, uncertainty sampling, and EER, and shows that impact sampling produces the best results.

fected by the additional hypothetical labels. One can keep track of such examples using k-d trees and geometric reasoning about the domain.

## Behavior of Impact Sampling

We first return to the example of Figure 1, to verify that `impactEXP` indeed solves the problems that traditional active learning algorithms encounter when applied to re-active learning . As a given point is labeled more and more, the expected impact of that point goes to 0, because the probability that the aggregated label $f(L_{x_i})$ changes with an additional label goes to 0 as the number of labels for example $x_i$ goes to infinity. Thus, impact sampling considers both classifier and label uncertainty, and will pick different points that have not yet been labeled many times, avoiding the infinite loops that starve $\texttt{US}_{\mathcal{X}}$ and EER.

While impact sampling and uncertainty sampling are ostensibly optimizing for two different objectives, we now show that, surprisingly, impact sampling is a generalization of uncertainty sampling to the re-active learning setting. Specifically, in some *noiseless* learning problems (where relabeling is unnecessary) `impactEXP` reduces to $\texttt{US}_{\mathcal{X}_\mathcal{U}}$. Yet when relabeling is allowed, `impactEXP` behaves quite differently than $\texttt{US}_{\mathcal{X}}$, the extension of $\texttt{US}_{\mathcal{X}_\mathcal{U}}$ that *can* relabel.

For example, consider the following setting. Let $\mathcal{P}$ be the class of 1-D threshold noiseless learning problems, a popular class in the active learning literature (Dasgupta 2004; 2005): The goal is to learn a threshold $t^*$ on $\mathcal{X} = [a, b], a, b \in \mathbb{R}, a < b$, where $\mathcal{D}$ is uniformly distributed on $\mathcal{X}$, and the data are linearly separable. We assume that active learning methods subsample from $\mathbb{R}$ first to get a finite set of candidate points. Suppose we are learning a max-margin classifier that outputs probability predictions $P_{\mathcal{A}}(h^*(x_i) = 0)$ and $P_{\mathcal{A}}(h^*(x_i) = 1)$ that are *monotonic*, i.e., for all $x_1, x_2 \in \mathcal{X}$, $x_1 > x_2$ if and only if $P_{\mathcal{A}}(h^*(x_1) = 1) > P_{\mathcal{A}}(h^*(x_2) = 1)$ and $P_{\mathcal{A}}(h^*(x_1) = 0) < P_{\mathcal{A}}(h^*(x_2) = 0)$, or $x_1 > x_2$ if and only if $P_{\mathcal{A}}(h^*(x_1) = 1) < P_{\mathcal{A}}(h^*(x_2) = 1)$ and $P_{\mathcal{A}}(h^*(x_1) = 0) > P_{\mathcal{A}}(h^*(x_2) = 0)$. Max-margin classifiers like SVMs can easily output monotonic probability predictions (e.g., by normalizing the distance to the hyperplane).

**Theorem 2.** *In the noiseless learning problem $\mathcal{P}$ (with no relabeling), $x_i$ is the point chosen by $\texttt{US}_{\mathcal{X}_\mathcal{U}}$ if and only if $x_i$ is the point chosen by `impactEXP`.*

See the Supplementary Materials for the proof of this theorem, as well as more general conditions for when impact sampling reduces to uncertainty sampling.

## Experiments

We now present empirical experiments on both synthetic and real-world datasets to compare the performances of $\texttt{US}_{\mathcal{X}}^{\alpha}$, $\texttt{US}_{\mathcal{X}_U}^{j/k}$, and impact sampling. As baselines, we include EER and $\texttt{US}_{\mathcal{X}_U}$ (uncertainty sampling *without* relabeling). We do *not* show results for $\texttt{US}_{\mathcal{X}}$, uncertainty sampling *with* relabeling, since it performs extremely poorly as explained earlier.

We begin with a synthetic domain containing two random Gaussian clusters, which correspond to two classes, and train using L2-regularized logistic regression. We generate a dataset by randomly picking two means, $\mu_1, \mu_2 \in [0, 1]^z$, and two corresponding covariance matrices $\Sigma_1, \Sigma_2 \in [0, 1]^{z \times z}$. We ensure a pool of 1,000 examples for each Gaussian cluster (class) exists at every timestep, in order to simulate an infinite pool of examples from which the algorithms may choose. All experiments are averaged over 250 random datasets (all figures show faded 95% confidence intervals, most of which are extremely small). We vary the number of features among $z \in \{10, 30, 50, 70, 90\}$. We seed training with 50 examples, use a total budget of 1,000, and test on 300 held-out examples. Note that our budget is much larger than what is common in the active learning literature, with many works experimenting with budgets 10-40 times smaller than ours (e.g., (Dasgupta and Hsu 2008; Golovin, Krause, and Ray 2010)). We use a larger budget for two reasons. First, to make our experiments more realistic than prior work, and second, to account for slower convergence due to noise. We assume the *classification noise model* (Angluin and Laird 1988): each label is independently flipped from the true label, $h^*(x)$, with probability

0.25. We assume that label accuracy is known and use majority vote for $f$, the label aggregation function.

Figure 2 shows the performance of several strategies when setting the number of features $z = 90$. Figure 2(a) compares $\texttt{US}^{\alpha}_{\mathcal{X}}$ with $\alpha \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$. We find that $\alpha = 0.1$ does the best, and that as we increase $\alpha$ from 0.1, the performance of the learned classifier drops, though not drastically. Although not shown, we also experiment with $\texttt{US}^{j/k}_{\mathcal{X}_U}$, varying $j/k \in \{1/1, 2/3, 3/5, 4/7, 5/9\}$. We see that performance improves as $j/k$ increases from $1/1$ to $3/5$ but then decreases as we continue to increase the amount of relabeling redundancy. Different $\alpha$ and $j/k$ work better with different number of features, and the best hyperparameter for a given number of features is difficult to predict.

Figure 2(b) shows the effect of introducing optimism and pseudo-lookahead into impact sampling. We notice that these methods are able to substantially increase performance. `impactOPT(2)` outperforms `impactEXP(2)`, and `impactPLOPT(2)` performs significantly better than `impactOPT(2)`. We also see that allowing impact sampling to choose from a larger pool of candidate examples (`impactPLOPT(7)`) improves performance over the 2-example version (`impactPLOPT(2)`). Henceforth, in subsequent figures we show only the performance of our most robust impact sampling algorithm, `impactPLOPT(7)`.

Although also not shown, we also investigate the effect of applying optimism and pseudo-lookahead to EER. We find that although such techniques help make some substantial initial gains, all versions of EER get caught by infinite loops and performance suffers due to starvation.

Figure 2(c) compares impact sampling to uncertainty sampling and EER. We observe that `impactPLOPT(7)` significantly outperforms all other methods ($p < 0.001$ using a Welch's t-test on the average accuracies at the maximum budget). Although not shown, we find that even `impactEXP(2)`, the weakest impact sampling strategy, strictly dominates vanilla $\texttt{US}_{\mathcal{X}_U}$. The results for $z \in \{30, 50, 70\}$ look very similar. However, when $z = 10$, all strategies perform about the same, because the learning problem is too easy. We also repeated these experiments using synthetic datasets with three Gaussian clusters instead of two, where one class contains a mixture of two clusters, and found extremely similar results.

Figure 3 compares impact sampling against uncertainty sampling on datasets from the UCI Machine Learning Repository (Bache and Lichman 2013) with synthetically-generated labels. We use a budget that is equal to half of the size of the dataset, and randomly generate a training set of 70% of the examples, a held-out set of 15% of the examples, and a test set of the remaining 15% of the examples. We again present results averaged over 250 of these randomizations and display 95% confidence intervals. In both datasets, `impactPLOPT(7)` is substantially and statistically significantly better than $\texttt{US}_{\mathcal{X}_U}$ ($p < 0.001$).

Finally, we investigate how well our methods work on real-world datasets with real human-generated labels. In the following experiments, we use F-score as the metric because of high skew. Whenever a strategy requests a label, we sample from the set of available annotations. Unlike the previous experiments, here we do *not* assume knowledge of label accuracy. To make this relaxation, first, we always apply optimism to all examples that we consider, instead of only when examples will be relabeled. Second, instead of including 5 points returned by $\texttt{US}^{\alpha}_{\mathcal{X}}$ among the subset of 7 from which impact sampling chooses, we include 5 random points.

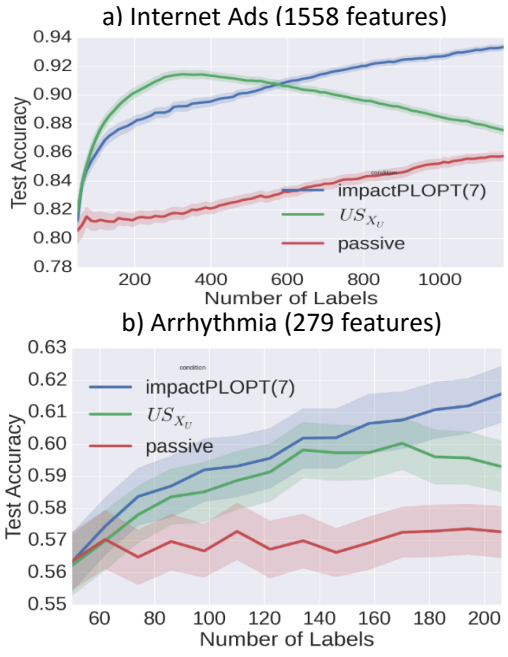


Figure 3: Comparison of impact sampling versus uncertainty sampling on real-world datasets, Internet Ads and Arrhythmia, using simulated labels modeling annotaters whose accuracy is 0.75. Impact sampling shows significant gains.

We first consider a popular dataset (Kamar, Hacker, and Horvitz 2012; Lintott et al. 2011; 2008) from Galaxy Zoo, a website on which volunteer workers contribute annotations for images of galaxies gathered by the Sloan Digital Sky Survey. Each image has 62 features obtained from machine vision analysis, and workers are asked to label each galaxy as "elliptical," "spiral," or "other." Each galaxy is labeled by at least 10 workers. We use a small version of the dataset containing 3937 galaxies, and train a logistic regression classifier to predict whether a galaxy is elliptical. About 12.8% of galaxies are in the positive class. Figure 4a shows that `impactPLOPT(7)` performs well against the baselines, but the difference is not statistically significant.

We also consider a dataset gathered for the task of relation extraction. Given a sentence, e.g., "Barack Obama was born in Hawaii," two entities in that sentence, "Barack Obama" and "Hawaii," and a relation, "*born-in*," the goal is to determine whether the given sentence expresses the given relation between the two entities. In this example, the sentence does express the relation that Barack Obama was born in Hawaii. The dataset contains 2000 sentences, each la-

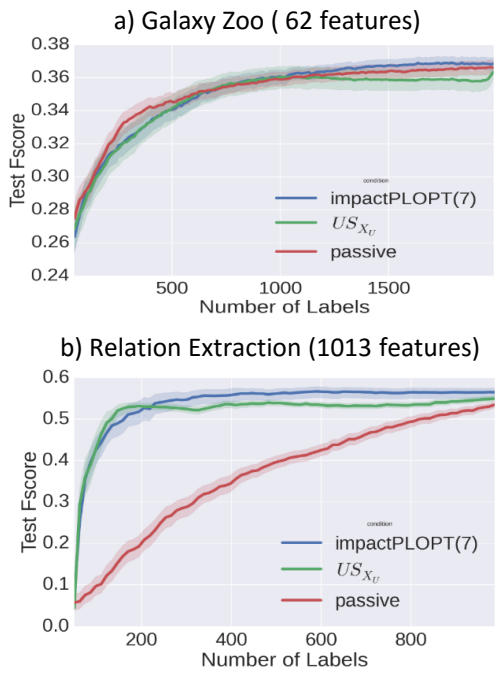


Figure 4: Impact sampling performs well against uncertainty sampling on the real-world datasets with real human-generated labels.

beled by 10 Amazon Mechanical Turk workers with at least a 95% approval rating. Each example sentence is described by 1013 features. We train a logistic regression classifier to predict whether an example expresses the "*born-in*" relation. Positive examples comprise about 7.4% of the dataset. Figure 4b shows that `impactPLOPT(7)` makes significant gains ($p < 0.01$) against the baselines.

## Discussion and Future Work

Our experiments show that impact sampling trains better classifiers than previous active learning algorithms *given the same labeling budget*. However, this assessment does not consider the cost of computational resources or time. One drawback of our method, shared with active learning techniques such as expected error reduction (Roy and McCallum 2001; Kapoor, Horvitz, and Basu 2007), is the high computational cost of picking the next data point to (re)label. This computational burden stems from impact sampling's need to retrain a classifier twice per candidate data point. In some cases this computational load may be prohibitive, especially if the training cost of the classifier is high, such as for multilayer perceptrons or non-linear SVMs or in problems with high dimensional feature spaces. These cases raise the question of whether or not re-active learning is truly useful in practice.

For example, in our final experiment on relation extraction, impact sampling takes over 7 hours[1] to choose the set of 1000 examples to (re)label. In other words, impact sampling takes, on average, about 26 seconds to compute the best example to send to a human, who would likely take about the same time to label the example. In contrast, uncertainty sampling and random sampling are much faster, taking approximately 210 and 70 seconds, respectively, to select their 1000 examples.

Note, however, that there are many ways to increase the speed of impact sampling. As we have discussed and shown, instead of choosing an example to (re)label from all of $\mathcal{X}$, choosing among a small subset of $\mathcal{X}$ can result in large computational savings and still be very effective. And although in our implementation we used a simple logistic regression classifier that was retrained from scratch after each conjectured label, when used with a classifier that has a mechanism for incremental training (e.g., stochastic gradient descent initialized with the previous solution or incremental decision tree induction), impact sampling would likely run much more quickly and scale to even more complex learning problems. If one is faced with a problem too large for impact sampling, we suggest using $\texttt{US}_{\mathcal{X}_U}$, uncertainty sampling with no relabeling, but note that one will likely require significantly more human annotation than needed by impact sampling to achieve the same learner accuracy.

Finally, we note that through our experiments, we have shown the robustness of impact sampling, but like with all active learning techniques, there exist datasets in which our methods do not perform well. Further work is necessary to understand the properties of learning problems that affect the performance of re-active learning algorithms.

## Conclusion

Our paper tackles the problem of re-active learning, a generalization of active learning that explores the tradeoff between decreasing the noise of the training set via relabeling and increasing the size of the (noisier) training set by labeling new examples. We introduce two re-active learning algorithms: an extension of uncertainty sampling, and a novel of class of impact sampling algorithms. We characterize their theoretical properties and investigate interrelationships among these algorithms. We find that impact sampling is equivalent to uncertainty sampling for some noiseless cases, but provides significant benefit for the case of noisy annotations. We empirically demonstrate the effectiveness of impact sampling on synthetic domains, real-world domains with synthetic labels, and real-world domains with real human-generated labels.

## Acknowledgments

We thank Jonathan Bragg, Gagan Bansal, Ece Kamar, and Eric Horvitz for helpful discussions. Xiao Ling contributed a key insight used in our proofs. We thank the anonymous reviewers for their useful comments. This work was supported by NSF grant IIS-1420667, ONR grant N00014-12-1-0211, the WRF/Cable Professorship, a gift from Google,

[1]Recall that this domain has 1013 features. Experiments are programmed in Python using an Intel Xeon E7-4850-v2 processor (2.3 GHz, 24M Cache) with 512 GB of RAM.

a language understanding and knowledge discovery focused award from Google, and a research grant from KISTI.

## References

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

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

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

Bragg, J.; Mausam; and Weld, D. S. 2014. Parallel task routing for crowdsourcing. In *HCOMP*.

Chen, X.; Bennett, P. N.; Collins-Thompson, K.; and Horvitz, E. 2013. Pairwise ranking aggregation in a crowd-sourced setting. In *WSDM*.

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

Dasgupta, S., and Hsu, D. 2008. Hierarchical sampling for active learning. In *ICML*.

Dasgupta, S. 2004. Analysis of a greedy active learning strategy. In *NIPS*.

Dasgupta, S. 2005. Coarse sample complexity bounds for active learning. In *NIPS*.

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*.

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

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.

Kääriäinen, M. 2006. Active learning in the non-realizable case. In *ALT*.

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

Kamar, E.; Hacker, S.; and Horvitz, E. 2012. Combining human and machine intelligence in large-scale crowdsourcing. In *AAMAS*.

Kapoor, A.; Horvitz, E.; and Basu, S. 2007. Selective supervision: Guiding supervised learning with decision-theoretic active learning. In *IJCAI*.

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

Krempl, G.; Kottke, D.; and Lemaire, V. 2015. Optimised probabilistic active learning. *Machine Learning* 100(2):449–476.

LDC. 2015. Linguistic Data Consortium: Annotation Overview, `ldc.upenn.edu/communications/data-sheets/annotation-overview`.

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

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*.

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

Lintott, C. J.; Schawinski, K.; Slosar, A.; Land, K.; Bamford, S.; Thomas, D.; Raddick, M. J.; Nichol, R. C.; Szalay, A.; Andreescu, D.; Murray, P.; and Vandenberg, J. 2008. Galaxy zoo: morphologies derived from visual inspection of galaxies from the sloan digital sky survey. *Monthly Notices of the Royal Astronomical Society* 389(3):1179–1189.

Lintott, C.; Schawinski, K.; Bamford, S.; Slosar, A.; Land, K.; Thomas, D.; Edmondson, E.; Masters, K.; Nichol, R. C.; Raddick, M. J.; Szalay, A.; Andreescu, D.; Murray, P.; and Vandenberg, J. 2011. Galaxy zoo 1: data release of morphological classifications for nearly 900 000 galaxies. *Monthly Notices of the Royal Astronomical Society* 410(1):166–178.

Roy, N., and McCallum, A. 2001. Toward optimal active learning through sampling estimation of error reduction. In *ICML*.

Settles, B.; Craven, M.; and Ray, S. 2007. Multiple-instance active learning. In *NIPS*.

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

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*.

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

Werling, K.; Chaganty, A.; Liang, P.; and Manning, C. D. 2015. On-the-job learning with bayesian decision theory. 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*.

Zhang, C., and Chaudhuri, K. 2015. Active learning from weak and strong labelers. In *NIPS*.