Active Learning for Cost-Sensitive Classification

Akshay Krishnamurthy [1] **Alekh Agarwal** [2] **Tzu-Kuo Huang** [3] **Hal Daumé III** [4] **John Langford** [2]

Abstract

We design an active learning algorithm for cost-sensitive multiclass classification: problems where different errors have different costs. Our algorithm, COAL, makes predictions by regressing to each label's cost and predicting the smallest. On a new example, it uses a set of regressors that perform well on past data to estimate possible costs for each label. It queries only the labels that *could be* the best, ignoring the sure losers. We prove COAL can be efficiently implemented for any regression family that admits squared loss optimization; it also enjoys strong guarantees with respect to predictive performance and labeling effort. Our experiment with COAL show significant improvements in labeling effort and test cost over passive and active baselines.

1. Introduction

The field of active learning studies how to efficiently elicit relevant information so learning algorithms can make good decisions. Almost all active learning algorithms are designed for binary classification problems, leading to the natural question: How can active learning address more complex prediction problems? Multiclass and importance-weighted classification require only minor modifications but we know of no active learning algorithms that enjoy theoretical guarantees for more complex problems.

One such problem is cost-sensitive multiclass classification (CSMC). In CSMC with K classes, passive learners receive input examples x and cost vectors $c \in \mathbb{R}^K$, where $c(y)$ is the cost of predicting label y on x.[1] A natural design for an *active* CSMC learner then is to adaptively query the costs of only a (possibly empty) subset of labels on each x. Since measuring label complexity is more nuanced in CSMC (e.g., is it more expensive to query three costs on a single example or one cost on three examples?), we track both the number of examples for which at least one cost is queried, along with the total number of cost queries issued. The first corresponds to a fixed human effort for inspecting x. The second captures the additional effort for judging the cost of each prediction, which depends on the number of labels queried. (By querying a label, we mean querying the cost of predicting that label given an example.)

In this setup, we develop a new active learning algorithm for CSMC called Cost Overlapped Active Learning (COAL). COAL assumes access to a set of regression functions, and, when processing an example x, it uses the functions with good past performance to compute the range of possible costs that each label might take. Naturally, COAL only queries labels with large cost range, but furthermore, it only queries y's that could possibly have the smallest cost, avoiding the uncertain, but surely suboptimal labels. The key algorithmic innovation is an efficient way to compute the cost range realized by good regressors. This computation, and COAL as a whole, only requires that the regression set admits efficient squared loss optimization, in contrast with prior algorithms that require 0/1 loss optimization (Beygelzimer et al., 2009; Hanneke, 2014).

Among our results, we prove that when processing n (unlabeled) examples with K classes and N regressors,

1. The algorithm needs to solve $\mathcal{O}(Kn^2 \log n)$ regression problems over the function class (Cor. 2), which can be done in polynomial time for convex regression sets.
2. With no assumptions on the noise in the problem, the algorithm achieves generalization error $\tilde{\mathcal{O}}(\sqrt{K \ln N/n})$ and requests $\tilde{\mathcal{O}}(n\theta_2\sqrt{K \ln N})$ costs from $\tilde{\mathcal{O}}(n\theta_1\sqrt{K \ln N})$ examples (Thms. 3 and 5) where θ_1, θ_2 are the disagreement coefficients (Def. 1)[2]. The worst case offers minimal improvement over passive learning, akin to binary classification.
3. With a favorable noise assumption (As. 2), the algorithm achieves generalization error $\tilde{\mathcal{O}}(K \ln N/n)$ while requesting $\tilde{\mathcal{O}}(Kc^{1/\beta}n^{\beta}\theta_2 \ln N)$ labels from $\tilde{\mathcal{O}}(c^{1/\beta}n^{\beta}\theta_1 K \ln N)$ examples (Cor. 4, Thm. 6), where

[1]University of Massachusetts, Amherst, MA [2]Microsoft Research, New York, NY [3]Uber Advanced Technology Center, Pittsburgh, PA [4]University of Maryland, College Park, MD. Correspondence to: Akshay Krishnamurthy <akshay@cs.umass.edu>.

Proceedings of the 34^{th} International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017.

[1]Cost here refers to prediction cost and not labeling effort or the cost of acquiring different labels.

[2]$\tilde{\mathcal{O}}(\cdot)$ suppresses logarithmic dependence on n and K.

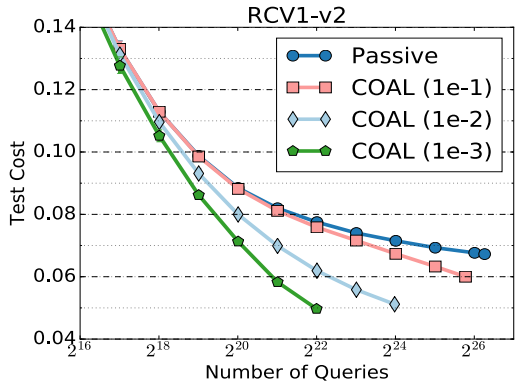

Figure 1. Empirical evaluation of COAL on Reuters text categorization dataset. Active learning achieves *better* test cost than passive, with a factor of 16 fewer queries. See Section 6 for details.

$\beta \in (0,1)$ is a safety parameter and c is a constant.

We also discuss some intuitive examples highlighting the benefits of using COAL.

CSMC provides a more expressive language for success and failure than multiclass classification, which allows algorithms to better trade-off errors and broadens potential applications. For example, CSMC can naturally express partial failure in hierarchical classification (Silla Jr. & Freitas, 2011). Experimentally, we show that COAL substantially outperforms the passive learning baseline with orders of magnitude savings in the labeling effort on a number of hierarchical classification datasets (see Fig. 1 for comparison between passive learning and COAL on Reuters text categorization).

CSMC also forms the basis of learning to avoid cascading failures in joint prediction tasks (Daumé III et al., 2009; Ross & Bagnell, 2014; Chang et al., 2015) like structured prediction and reinforcement learning. As our second application, we consider learning to search algorithms for joint (or structured) prediction, which operate by a reduction to CSMC. In this reduction, evaluating the cost of a class often involves a computationally expensive "roll-out," so using an active learning algorithm inside such a (passive) joint prediction method can lead to significant computational savings. We show that using COAL within the AGGRAVATE algorithm (Ross & Bagnell, 2014; Chang et al., 2015) reduces the number of roll-outs by a factor of $\frac{1}{4}$ to $\frac{3}{4}$ on several joint prediction tasks.

Related Work. Active learning is a thriving research area with many theoretical and empirical studies. We recommend the survey of Settles (2012) for an overview of more empirical research. We focus here on theoretical results.

Castro & Nowak (2008) study active learning with nonparametric decision sets, while Balcan et al. (2007); Balcan & Long (2013) focus on linear representations under distributional assumptions. The online learning community has also studied active learning of linear separators under adversarial assumptions (Cavallanti et al., 2011; Dekel et al., 2010; Orabona & Cesa-Bianchi, 2011; Agarwal, 2013).

Our work falls into the framework of *disagreement-based active learning*, which studies general hypothesis spaces typically in an agnostic setup (see Hanneke (2014) for an excellent survey). Existing results study binary classification, while our work generalizes to CSMC, assuming that we can accurately predict costs using regression functions from our class. The other main differences are that our query rule checks the range of predicted costs for a label, and we use a square loss oracle to search the version space.

In contrast, prior work either explicitly enumerates the version space (Balcan et al., 2006; Zhang & Chaudhuri, 2014) or uses a 0/1 loss *classification* oracle for the search (Dasgupta et al., 2007; Beygelzimer et al., 2009; 2010; Huang et al., 2015). In most instantiations, the oracle solves an NP-hard problem and so does not directly lead to an efficient algorithm, although practical implementations using heuristics are still quite effective. Our approach instead uses a squared-loss *regression* oracle, which can often be implemented efficiently via convex optimization and leads to a polynomial time algorithm.

Supervised learning oracles that solve NP-hard optimization problems in the worst case have been used in other problems including contextual bandits (Agarwal et al., 2014; Syrgkanis et al., 2016) and structured prediction (Daumé III et al., 2009). Thus we hope that our work can inspire new algorithms for these settings as well.

Lastly, we mention that square loss regression has been used to estimate costs for passive CSMC (Langford & Beygelzimer, 2005), but, to our knowledge, using a square loss oracle for active CSMC is new.

2. Problem Setting and Notations

We study cost-sensitive multiclass classification problems with K classes, where there is an instance space $\mathcal{X}$, a label space $Y = \{1, \ldots, K\}$, and a distribution $\mathcal{D}$ supported on $\mathcal{X} \times [0,1]^K$.[3] If $(x, c) \sim \mathcal{D}$, we refer to c as the *cost-vector*, where $c(y)$ is the cost of predicting $y \in Y$. A classifier $h : \mathcal{X} \to Y$ has expected cost $\mathbb{E}_{(x,c)\sim\mathcal{D}}[c(h(x))]$ and we aim to find a classifier with minimal expected cost.

Let $\mathcal{G} \triangleq \{g : \mathcal{X} \mapsto [0,1]\}$ denote a set of base regressors and let $\mathcal{F} \triangleq \mathcal{G}^K$ denote a set of vector regressors where the y^{th} coordinate of $f \in \mathcal{F}$ is written as $f(\cdot; y)$. The set of classifiers under consideration is $\mathcal{H} \triangleq \{h_f \mid f \in \mathcal{F}\}$

[3] In general, labels just serve as indices for the cost vector in CSMC, and the data distribution is over (x, c) pairs instead of (x, y) pairs as in binary and multiclass classification.

where each f defines a classifier $h_f : \mathcal{X} \mapsto Y$ by

$$h_f(x) \triangleq \operatorname*{argmin}_y f(x; y). \quad (1)$$

Given a set of examples and queried costs, we often restrict attention to regression functions that predict these costs well, and assess the uncertainty in their predictions given a new example x. For a subset of regressors $G \subset \mathcal{G}$, we measure uncertainty over possible cost values for x with

$$\gamma(x, G) \triangleq \underbrace{c_+(x, G)}_{\triangleq \max_{g \in G} g(x)} - \underbrace{c_-(x, G)}_{\triangleq \min_{g \in G} g(x)}. \quad (2)$$

For vector regressors $F \subset \mathcal{F}$, we define the *cost range* for a label y given x as $\gamma(x, y, F) \triangleq \gamma(x, G_F(y))$ where $G_F(y) \triangleq \{f(\cdot; y) \mid f \in F\}$ is the set of base regressors induced by F for y.

When using a set of regression functions for a classification task, it is natural to assume that the expected costs under $\mathcal{D}$ can be predicted well by some function in the set. This motivates the following realizability assumption.

Assumption 1 (Realizability)**.** *Define the Bayes-optimal regressor $f^\star$, which has $f^\star(x; y) = \mathbb{E}_c[c(y)|x], \forall x \in \mathcal{X}$ (with $\mathcal{D}(x) > 0$), $y \in Y$. We assume that $f^\star \in \mathcal{F}$.*

While $f^\star$ is always well defined, note that the cost itself may be noisy. In comparison with our assumption, the existence of a zero-cost classifier in $\mathcal{H}$ (which is often assumed in active learning) is stronger, while the existence of $h_{f^\star}$ in $\mathcal{H}$ is weaker but has not been leveraged in active learning.

In typical settings, the set $\mathcal{G}$ is extremely large, which introduces a computational challenge of managing this set. To address this challenge, we leverage existing algorithmic research on supervised learning and assume access to a regression oracle for $\mathcal{G}$. Given an importance-weighted dataset $D = \{x_i, c_i, w_i\}_{i=1}^n$ the regression oracle computes

$$\textsc{Oracle}(D) \in \operatorname*{argmin}_{g \in \mathcal{G}} \sum_{i=1}^n w_i (g(x_i) - c_i)^2. \quad (3)$$

In many cases this is a convex problem and can be solved efficiently. In the special case of linear functions, this is just least squares and can be computed in closed form.

To measure the labeling effort, we track the number of examples for which even a single cost is queried as well as the total number of queries. This bookkeeping captures settings where the editorial effort for inspecting an example is high, but each cost requires minimal further effort, as well as those where each cost requires substantial effort. Formally, we define $Q_i(y)$ to be the indicator that the algorithm queries label y on the i^{th} example and measure

$$L_1 \triangleq \sum_{i=1}^n \bigvee_y Q_i(y), \text{ and } L_2 \triangleq \sum_{i=1}^n \sum_y Q_i(y). \quad (4)$$

Algorithm 1 Cost Overlapped Active Learning (COAL)

1: Input: Regressors $\mathcal{G}$, failure probability $\delta \leq 1/e$, safety parameter $\beta \in (0, 1)$.
2: Set $\eta_i = 1/\sqrt{i}$, $\kappa = 80$, $\nu_n = \log(2n^2|\mathcal{G}|K/\delta)$.
3: Set $\Delta_i = \frac{\kappa \epsilon_{i-1}}{i-1}$, $\epsilon_i = \left(\frac{n}{i}\right)^\beta \nu_n$.
4: **for** $i = 1, 2, \ldots, n$ **do**
5: $\quad g_{i,y} \leftarrow \arg\min_{g \in \mathcal{G}} \widehat{R}_i(g; y)$. (See Eq. (5))
6: $\quad$ Define $f_i \leftarrow \{g_{i,y}\}_{y=1}^K$.
7: $\quad \mathcal{G}_i(y) \leftarrow \{g \in \mathcal{G} \mid \widehat{R}_i(g; y) \leq \widehat{R}_i(g_{i,y}; y) + \Delta_i\}$.
8: $\quad$ Receive new example x. $Q_i(y) \leftarrow 0, \forall y \in Y$.
9: $\quad$ **for** every $y \in Y$ **do**
10: $\quad\quad \widehat{c_+}(y) \leftarrow \textsc{MaxCost}((x, y), \Delta_i, \frac{\eta_i}{4\sqrt{3}}, \widehat{R}_i(\cdot; y))$.
11: $\quad\quad \widehat{c_-}(y) \leftarrow \textsc{MinCost}((x, y), \Delta_i, \frac{\eta_i}{4\sqrt{3}}, \widehat{R}_i(\cdot; y))$.
12: $\quad$ **end for**
13: $\quad Y' \leftarrow \{y \in Y \mid \widehat{c_-}(y) \leq \min_{y'} \widehat{c_+}(y')\}$.
14: $\quad$ **if** $|Y'| > 1$ **then**
15: $\quad\quad Q_i(y) \leftarrow 1$ if $y \in Y'$ and $\widehat{c_+}(y) - \widehat{c_-}(y) > \eta_i$.
16: $\quad$ **end if**
17: $\quad$ Query costs of each y with $Q_i(y) = 1$.
18: **end for**

3. Cost Overlapped Active Learning

The pseudocode for our algorithm, Cost Overlapped Active Learning (COAL), is given in Algorithm 1. Given an example x, COAL queries the costs of some of the labels y for x. These costs are chosen by (1) computing a set of good regression functions based on the past data (i.e., the version space), (2) computing the range of predictions achievable by these functions for each y, and (3) querying each y that could be the best label *and* has substantial uncertainty. We now detail each step.

To compute an approximate version space we first find the regression function that minimizes the empirical risk for each label y, which at round i is:

$$\widehat{R}_i(g; y) = \frac{1}{i-1} \sum_{j=1}^{i-1} (g(x_j) - c_j(y))^2 Q_j(y). \quad (5)$$

Recall that $Q_j(y)$ is the indicator that we query label y on the j^{th} example. Computing the minimizer requires one oracle call. We implicitly construct the version space $\mathcal{G}_i(y)$ in Line 7 as the regressors with low square loss regret to the empirical risk minimizer. The tolerance on this regret is Δ_i at round i, which depends on the safety parameter $\beta \in (0, 1)$ in the algorithm. When β is large, the tolerance is also large and the algorithm issues many queries. Conversely when β is small the algorithm is more aggressive. However, for any strictly positive β, the definition of Δ_i ensures that $f^\star(\cdot; y) \in \mathcal{G}_i(y)$ for all i, y.

COAL then computes the maximum and minimum costs predicted by the version space $\mathcal{G}_i(y)$ on the new example x. Since the true expected cost is $f^\star(x; y)$ and $f^\star(\cdot; y) \in$

$\mathcal{G}_i(y)$, these quantities serve as a confidence bound for this value. The computation is done by the MAXCOST and MINCOST subroutines which produce approximations to $c_+(x, \mathcal{G}_i(y))$ and $c_-(x, \mathcal{G}_i(y))$ (Eq. (2)) respectively.

Finally, using the predicted costs, COAL issues (possibly zero) queries. The algorithm queries any *non-dominated* label that has a large *cost range*, where a label is non-dominated if its estimated minimum cost is smaller than the smallest maximum cost (among all labels) and the cost range is the difference between the label's estimated maximum and minimum costs.

Intuitively, COAL queries the cost of every label which cannot be ruled out as having the smallest cost on x, but only if there is sufficient ambiguity about the actual value of the cost. The idea is that labels with little disagreement do not provide much information for further reducing the version space, since by construction all functions would suffer similar loss. Moreover, only the labels that could be the best need to be queried at all, since the cost-sensitive performance of a hypothesis h_f depends only on the label that it predicts. Hence, labels that are dominated or have small cost range need not be queried.

Similar query rules appear in prior works on binary and multiclass classification (Orabona & Cesa-Bianchi, 2011; Dekel et al., 2010; Agarwal, 2013), but specialized to linear representations. The key advantage of the linear case is that the set $\mathcal{G}_i(y)$ (formally, a different set with similar properties) along with $c_+(y)$ and $c_-(y)$ have closed form expressions, so that the algorithms are easily implemented. However, with a general set $\mathcal{G}$ and a regression oracle, computing these confidence intervals is less straightforward. We use the MAXCOST and MINCOST subroutines, and discuss this aspect of our algorithm next.

3.1. Efficient Computation of Cost Range

In this section, we describe the MAXCOST subroutine which uses the oracle to approximate the maximum cost on label y realized by $\mathcal{G}_i(y)$ (recall definition in Eq. (2)).[4]

Describing the algorithm requires some additional notation. Given the empirical risk functional $\widehat{R}(g; y)$ over a set of examples (we suppress the subscript as the number of examples is fixed here), we define a weighted risk functional incorporating a fresh unlabeled example x as

$$\widetilde{R}(g, w, c; y) = \widehat{R}(g; y) + w(g(x) - c)^2. \qquad (6)$$

Finding $\operatorname{argmin}_g \widetilde{R}(g, w, c; y)$ involves a single oracle call. We also define a set of near-optimal regressors

$$\mathcal{G}(\Delta; y) = \left\{ g \in \mathcal{G} \mid \widehat{R}(g; y) - \min_{g'} \widehat{R}(g'; y) \leq \Delta \right\}. \quad (7)$$

[4] MINCOST is similar and omitted due to space constraints.

Algorithm 2 MAXCOST

Input: (x, y), Δ, ϵ, risk functional $\widehat{R}(\cdot; y)$
$g_{\min} = \operatorname{argmin}_{g \in \mathcal{G}} \widehat{R}(g; y)$.
$\ell = 0, h = 1, c = 1$.
while $|h - \ell| \geq 2\sqrt{3}\epsilon$ **do**
  $g_c \leftarrow \operatorname{argmin}_{g \in \mathcal{G}} \widetilde{R}(g, \Delta/\epsilon^2, c; y)$ (see Eq. 6).
  If $g_c \in \mathcal{G}(\Delta; y)$ (see Eq. 7), output $g_c(x) + \epsilon$.
  $(g_l, g_h) \leftarrow$ BSEARCH$((x, y, c), \epsilon, \Delta, \widehat{R}(\cdot; y))$.
  If $g_h \in \mathcal{G}(4\Delta; y)$, output $g_h(x)$.
  Else $\ell \leftarrow \max\{g_l(x), \ell\}$, $h \leftarrow g_h(x)$, $c \leftarrow \frac{h+\ell}{2}$.
end while
return c.

Algorithm 3 BINARYSEARCH(BSEARCH)

Input: (x, y, c), ϵ, Δ, risk functional $\widehat{R}(\cdot; y)$.
$w_{1,\ell} = 0, w_{1,h} = \Delta/\epsilon^2, t = 1$.
while $|w_{t,\ell} - w_{t,h}| \geq 2\Delta$ **do**
  $w_t \leftarrow \frac{w_{t,\ell} + w_{t,h}}{2}$, $g_t = \operatorname{argmin}_{g \in \mathcal{G}} \widetilde{R}(g, w_t, c; y)$.
  If $g_t \in \mathcal{G}(\Delta; y)$, $w_{t+1,\ell} \leftarrow w_t, w_{t+1,h} \leftarrow w_{t,h}$.
  Else $w_{t+1,\ell} \leftarrow w_{t,\ell}$, $w_{t+1,h} \leftarrow w_t$.
  $t \leftarrow t + 1$.
end while
return $g_\ell = \operatorname{argmin}_g \widetilde{R}(g, w_{t,\ell}, c; y)$, and $g_h = \operatorname{argmin}_g \widetilde{R}(g, w_{t,h}, c; y)$.

Thus at round i, the set $\mathcal{G}_i(y)$ in COAL is equivalent to $\mathcal{G}(\Delta_i; y)$, although we will use different radii here.

The algorithm for the maximum cost approximation, displayed in Algorithm 2, is based on a form of binary search. When invoked with a radius parameter Δ, the algorithm maintains an interval $[\ell, h]$ that contains $c_+(x, \mathcal{G}(\Delta; y))$ and uses a binary search to refine the interval. Using a fixed cost c and starting with some initial weight w, at each iteration, the binary search computes $\operatorname{argmin}_g \widetilde{R}(g, w, c; y)$ and verifies if the resulting regressor belongs to $\mathcal{G}(\Delta; y)$. If it does, it increases w, and otherwise it shrinks w. Once a termination criteria is reached, the BINARYSEARCH routine outputs two regressors (g_ℓ, g_h) that provide new upper and lower bounds on $c_+(x, \mathcal{G}(\Delta; y))$. The MAXCOST routine terminates and outputs $g_h(x)$ if it has reasonable empirical regret. Otherwise, it updates parameters for the next binary search based on $g_\ell(x), g_h(x)$.

Our main algorithmic result guarantees that this procedure produces an adequate approximation to $c_+(x, \mathcal{G}(\Delta; y))$ without requiring too many oracle calls.

Theorem 1. *For any (x, y), Δ, and ϵ, the* MAXCOST *algorithm outputs $\hat{c}$ satisfying*

$$c_+(x, \mathcal{G}(\Delta; y)) \leq \hat{c} \leq c_+(x, \mathcal{G}(4\Delta; y)) + \sqrt{3}\epsilon.$$

Further, the algorithm uses $\mathcal{O}(\epsilon^{-2} \log(1/\epsilon))$ oracle calls.

An immediate consequence of the theorem is a bound on the oracle complexity of COAL.

Corollary 2. *Over the course of n examples,* COAL *makes $O(Kn^2\log(n))$ calls to the square loss oracle.*

Thus COAL can be implemented in polynomial time for any set $\mathcal{G}$ that admits efficient square loss optimization. However, in practice, the number of oracle calls and the oracle itself are too computationally demanding to scale to larger problems. Our implementation alleviates this with an alternative heuristic approximation based on a sensitivity analysis of the oracle, which we detail in Section 6.

4. Generalization Analysis

In this section, we derive generalization guarantees for COAL. Our analysis assumes that the regressor set $\mathcal{G}$ is large, but finite. We study two different settings: one with minimal assumptions and one low-noise setting.

Our low-noise assumption is related to the Massart noise condition (Massart & Nédélec, 2006), which in binary classification posits that the Bayes optimal predictor is bounded away from $1/2$ for all x. Our condition generalizes this to CSMC and posits that the expected cost of the best label is separated from the expected cost of all other labels.

Assumption 2. *A distribution $\mathcal{D}$ supported over (x, c) pairs satisfies the* Massart noise condition*, if there exists $\tau > 0$ such that for all x (with $\mathcal{D}(x) > 0$),*

$$f^\star(x; y^\star(x)) \leq \min_{y \neq y^\star(x)} f^\star(x; y) - \tau,$$

where $y^\star(x) = \mathit{argmin}_y f^\star(x; y)$.

The Massart noise condition describes favorable prediction problems that lead to sharper generalization and label complexity bounds for COAL. COAL can also be analyzed under a milder assumption inspired by the Tsybakov noise condition, an analysis that we defer to an extended version.

Our results depend on the noise level in the problem, which we define using the following quantity, given any $\zeta > 0$.

$$P_\zeta \triangleq \Pr_{x\sim\mathcal{D}}[\min_{y\neq y^\star(x)} f^\star(x; y) - f^\star(x; y^\star(x)) \leq \zeta]. \quad (8)$$

P_ζ describes the probability that the expected cost of the best label, which is $y^\star(x)$, is close to the expected cost of the second best label. When P_ζ is small for large ζ the labels are well-separated so learning is easier. For instance, under a Massart condition $P_\zeta = 0$ for all $\zeta \leq \tau$.

We now state our generalization guarantee.

Theorem 3. *For any $\delta < 1/e$, for all $i \in [n]$, with probability at least $1-\delta$, we have*

$$\mathbb{E}_{x,c}[c(h_{f_{i+1}}(x)) - c(h_{f^\star}(x))] \leq \min_{\zeta>0}\left\{\zeta P_\zeta + \frac{2\kappa K\nu_n}{\zeta i}\right\},$$

where $\kappa = 80$, $\nu_n = \log\left(\frac{2n^2|\mathcal{G}|K}{\delta}\right)$, f_i is as defined in Line 6 of Algorithm 1, and h_{f_i} is defined in Equation (1).

In the worst case, we bound P_ζ by 1 and optimize for ζ to obtain an $\tilde{\mathcal{O}}(\sqrt{K\log(|\mathcal{G}|/\delta)/i})$ bound after i samples. To compare, the standard generalization bound is $\tilde{\mathcal{O}}(\sqrt{\log(|\mathcal{F}|/\delta)/i})$ (Littlestone & Warmuth, 1989), which agrees with our bound since $|\mathcal{F}| = |\mathcal{G}|^K$ in our case.

However, since the bound captures the difficulty of the CSMC problem as measured by P_ζ, we can obtain a sharper result under Assumption 2 by setting $\zeta = \tau$.

Corollary 4. *Under Assumption 2, for any $\delta < 1/e$, for all $i \in [n]$, with probability at least $1-\delta$, we have*

$$\mathbb{E}_{x,c}[c(h_{f_{i+1}}(x)) - c(h_{f^\star}(x))] \leq \frac{2\kappa K\nu_n}{i\tau}.$$

Thus, Massart-type conditions lead to a faster $\tilde{\mathcal{O}}(1/n)$ convergence rate. This agrees with the literature on active learning for classification (Massart & Nédélec, 2006) and can be viewed as a generalization to CSMC. Importantly, both generalization bounds recover the optimal rates and are independent of the safety parameter β.

5. Label Complexity Analysis

Without distributional assumptions, the label complexity of COAL can be $\mathcal{O}(n)$, just as in the binary classification case, since there may always be confusing labels that force querying. In line with prior work, we introduce two **disagreement coefficients** that characterize favorable distributional properties. We first define a set of good classifiers, the *cost-sensitive regret* ball:

$$\mathcal{F}_{\text{csr}}(r) = \left\{f \in \mathcal{F} \;\middle|\; \mathbb{E}\left[c(h_f(x)) - c(h_{f^\star}(x))\right] \leq r\right\}.$$

We may now define the disagreement coefficients.

Definition 1 (Disagreement coefficients)**.** *Define*

$$\gamma_r(x, y) = \gamma(x, y, \mathcal{F}_{csr}(r)), \;\; \textit{and}$$
$$\mathit{DIS}(r, y) = \{x \mid \exists f, f' \in \mathcal{F}_{csr}(r), h_f(x) = y \neq h_{f'}(x)\}.$$

Then the disagreement coefficients are defined as:

$$\theta_1 \triangleq \sup_{\eta_1, r>0} \frac{\eta_1}{r}\mathbb{P}\left(\exists y \mid \gamma_r(x, y) > \eta_1 \wedge x \in \mathit{DIS}(r, y)\right)$$
$$\theta_2 \triangleq \sup_{\eta_1, r>0} \frac{\eta_1}{r}\sum_y \mathbb{P}\left(\gamma_r(x, y) > \eta_1 \wedge x \in \mathit{DIS}(r, y)\right).$$

Intuitively, the conditions in both coefficients correspond to the checks on the *domination* and *cost range* of a label in Lines 13 and 15 of Algorithm 1. Specifically, when

$x \in \text{DIS}(r, y)$, there is confusion about whether y is the optimal label or not, and hence y is not dominated. The condition on $\gamma_r(x, y)$ additionally captures the fact that a small cost range provides little information, even when y is non-dominated. Collectively, the coefficients capture the probability of an example x where the good classifiers disagree substantially on x in both predicted costs and labels. Importantly, the notion of good classifiers is via the algorithm-independent set $\mathcal{F}_{\text{csr}}(r)$, and is only a property of $\mathcal{F}$ and $\mathcal{D}$.

The definition is a natural adaptation from binary classification (Hanneke, 2014), where a similar disagreement region to $\text{DIS}(r, y)$ is used. Our definition asks for confusion about the optimality of a specific label y, which provides more detailed information about the cost-structure than simply asking for any confusion among the good classifiers. The $1/r$ scaling leads to bounded coefficients in many examples (Hanneke, 2014), and we also scale by the cost range parameter η_1, so that the favorable settings for active learning can be concisely expressed as having θ_1, θ_2 bounded, as opposed to a complex function of η_1.

The next two results bound the labeling effort (Eq. (4)) in the high noise and low noise cases respectively. The low noise assumption enables a significantly sharper bound.

Theorem 5. *With probability at least $1-2\delta$, the label complexity of the algorithm over n examples is bounded by,*

$$L_1 = \mathcal{O}\left((25)^{1/\beta}\left(n\theta_1\sqrt{K\nu_n} + \log(1/\delta)\right)\right)$$
$$L_2 = \mathcal{O}\left((25)^{1/\beta}\left(n\theta_2\sqrt{K\nu_n} + K\log(1/\delta)\right)\right),$$

where $\nu_n = \log\left(\frac{2n^2|\mathcal{G}|K}{\delta}\right)$.

Theorem 6. *Assume the Massart noise condition holds. With probability at least $1-2\delta$ the label complexity of the algorithm over n examples is at most,*

$$L_1 = \mathcal{O}\left(\frac{25^{1/\beta}}{\tau^2}\left(n^\beta K\log(n)\nu_n\theta_1 + \log(1/\delta)\right)\right)$$
$$L_2 = \mathcal{O}\left(\frac{25^{1/\beta}K}{\tau^2}\left(n^\beta\log(n)\nu_n\left[K\theta_1+\theta_2\right] + \log(1/\delta)\right)\right).$$

In the high-noise case, the bounds scales with $n\theta$ for the respective coefficients. This agrees with results in binary classification, where at best constant-factor savings over passive learning are possible. On the other hand, in the low noise case, the label complexity scales as $\tilde{\mathcal{O}}(c^{1/\beta}n^\beta\theta/\tau^2)$, which is a polynomial improvement over passive learning. However, the constant scales exponentially with $1/\beta$ so β should not be chosen to be too small.

Note that θ_2 can be much smaller than $K\theta_1$, as demonstrated through an example in the next section. In such cases, only a few labels are ever queried and the L_2 bound in the high noise case reflects this additional savings over passive learning. Unfortunately, under Massart-noise, the L_2 bound depends directly on $K\theta_1$, so that we do not benefit when $\theta_2 \ll K\theta_1$. This can be resolved by letting η_i depend on the noise level τ, but we prefer to use the more robust choice $\eta_i = 1/\sqrt{i}$ which still allows COAL to partially *adapt* to low noise and achieve low label complexity.

Unfortunately, comparing our bound to binary classification reveals suboptimality here. Under Massart noise and bounded coefficients, the label complexity for binary classification is typically $\log(n)/\tau^2$ which contrasts with our n^β/τ^2 rate. This loss in rate arises from setting $\beta > 0$. However, with $\beta = 0$, a suboptimal regressor that left the version space may re-enter at a later round and cause us to issue more queries, since we may not accumulate evidence against this regressor unless it is in the version space. Binary classification methods address this issue by hallucinating labels for unqueried examples (Dasgupta et al., 2007), but hallucinating costs does not seem applicable to CSMC since the only safe choice that avoids eliminating f^* appears to be $f^*(x; y)$, which is unknown. Our solution uses $\beta > 0$ to induce a shrinking radius so that bad regressors cannot re-enter the version space. However, to avoid eliminating $f^\star$, the initial radius Δ_1 must be larger than standard concentration arguments require, so the algorithm is conservative. Information-theoretically (by enumerating the version space), the logarithmic rate is possible in CSMC, but we do not know of efficient algorithms for this.

5.1. Two Examples

Our first example shows the benefits of using the domination criterion in querying, in addition to the cost range condition. Consider a problem under Assumption 2, where the optimal cost is predicted perfectly, the second best cost is τ worse and all the other costs are substantially worse, but with variability in the predictions. Since all classifiers predict the right label, we get $\theta_1 = \theta_2 = 0$, so our label complexity bound is $\mathcal{O}(1)$. Intuitively, since every regressor is certain of the optimal label and its cost, we actually make zero queries. On the other hand, all of the suboptimal labels have large cost ranges, so querying based solely on a cost range criteria leads to a large label complexity.

A related example demonstrates the improvement in our query rule over more naïve approaches where we query either no label or all labels, which is the natural generalization of query rules from multiclass classification (Agarwal, 2013). In the above example, if the best and second best labels are confused occasionally θ_1 may be large, but we expect $\theta_2 \ll K\theta_1$ since only the second best label can be confused with the best. Thus, the L_2 bound in Theorem 5 is a factor of K smaller than with a naïve query rule since COAL only queries the best and second best labels.

	K	n	feat		K	n	feat	ℓ
INet 20	20	38k	6k	POS	45	38k	40k	24
INet 40	40	71k	6k	NER	9	15k	15k	14
RCV1-v2	103	781k	47k	Wiki	9	132k	89k	25

Table 1. Dataset statistics (ℓ is the average sequence length).

6. Experiments

For computational efficiency, we implemented an approximate version of COAL using *online optimization*, based on online linear least-squares regression. The algorithm processes the data in one pass, and the idea is to (1) replace $g_{i,y}$, the ERM, with an approximation $g^o_{i,y}$ obtained by online updates, and (2) compute the minimum and maximum costs via a sensitivity analysis of the online update. Specifically, we define the a *sensitivity* value $s(x, c, g^o_{i,y}) \geq 0$, which is the derivative of the prediction on x as a function of the importance weight w, for the new example x and cost $c = 0$ or $c = 1$ (for c_- and c_+ respectively). Then we approximate c_- via $g^o_{i,y}(x) - \underline{w}^o \cdot s(x, 0, g^o_{i,y})$ where $\underline{w}^o$ is the largest weight w satisfying

$$w(g^o_{i,y}(x)^2 - (g^o_{i,y}(x) - ws(x, 0, g^o_{i,y}))^2) \leq \Delta_i,$$

and Δ_i is the radius used at round i. We similarly approximate the maximum cost. See Appendix A for details.

6.1. Simulated Active Learning

We performed simulated active learning experiments with three datasets. ImageNet 20 and 40 are sub-trees of the ImageNet hierarchy covering 20 and 40 most frequent classes, where each example has a single zero-cost label and the cost for incorrect labels is the tree-distance to the correct one. The feature vectors are the top layer of the Inception neural network (Szegedy et al., 2015). The third dataset, RCV1-v2 (Lewis et al., 2004), is a multilabel text-categorization dataset, which has 103 topic labels, organized as a tree with similar tree-distance cost structure as the ImageNet data. Some dataset statistics are in Table 1.

We compare our online version of COAL to passive online learning. We use the cost-sensitive one-against-all (CSOAA) implementation in Vowpal Wabbit[5], which performs online linear regression for each label separately. There are two tuning parameters in our implementation. First, instead of Δ_i, we set the radius of the version space to $\Delta'_i = \frac{\kappa \nu_{i-1}}{i-1}$ (i.e. $\beta = 0$ and the log factor $\nu_i = \log\left(\frac{2(i-1)^2|\mathcal{G}|K}{\delta}\right)$ scales with i) and instead tune the constant κ. This alternate "mellowness" parameter controls how aggressive the query strategy is. The second parameter is the learning rate used by online linear regression[6].

For each parameter setting and each dataset, we make one pass through the training set and check the test cost (which is just the normalized expected cost) of the model every doubling number of queries. We repeat this on 100 random permutations of the training data and plot the results in Figures 1 and 2. For each mellowness, we show the results of the best learning rate, which maximizes a notion of AUC that reflects the trade-off between test cost and number of queries (see Eq. (11) in Appendix A).

The figures show, for each dataset and mellowness, the number of queries against the median test cost along with bars extending from the 15th to 85th quantile. Overall, COAL achieves a better trade-off between performance and queries. With proper mellowness parameter, active learning achieves similar test cost as passive learning with a factor of 8 to 32 less queries. On ImageNet 40 and RCV1-v2 (recall Figure 1), active learning achieves *better* test cost with a factor of 16 less queries. On RCV1-v2, COAL queries like passive up to around $256k$ queries, since the data is very sparse, and linear regression has the property that the cost range is maximal when an example has a new unseen feature. Once COAL sees all features a few times, it queries much more efficiently than passive. Note that these plots correspond to the label complexity L_2, with similar results for L_1 in Appendix A.3.

While not always the best, we recommend a mellowness setting of 0.01 as it achieves reasonable performance on all three datasets. This is also confirmed by the learning-to-search experiments, which we discuss in the next section.

We also compare COAL with two active learning baselines. Both algorithms differ from COAL only in their query rule. ALLORNONE queries either all labels or no labels using both domination and cost-range conditions and is an adaptation of existing multiclass active learners (Agarwal, 2013). NODOM just uses the cost-range condition, inspired by active regression (Castro et al., 2005). The results for ImageNet 40 are displayed in Figure 2 (See also Appendix A.3), where we use the AUC strategy to choose the learning rate. We choose the mellowness by visual inspection for the baselines and use 0.01 for COAL[7]. COAL substantially outperforms both baselines, which provide minimal improvement over passive learning.

6.2. Learning to Search

We also experiment with COAL as the base leaner in *learning-to-search* (Daumé III et al., 2009; Chang et al., 2015), which reduces joint prediction problems to CSMC. Here, a joint prediction example defines a search space,

[5] http://hunch.net/~vw

[6] We use the default online learning algorithm in Vowpal Wabbit, which is a scale-free (Ross et al., 2013) importance weight invariant (Karampatziakis & Langford, 2011) form of AdaGrad (Duchi et al., 2010).

[7] We use 0.01 for ALLORNONE and 10^{-3} for NODOM.

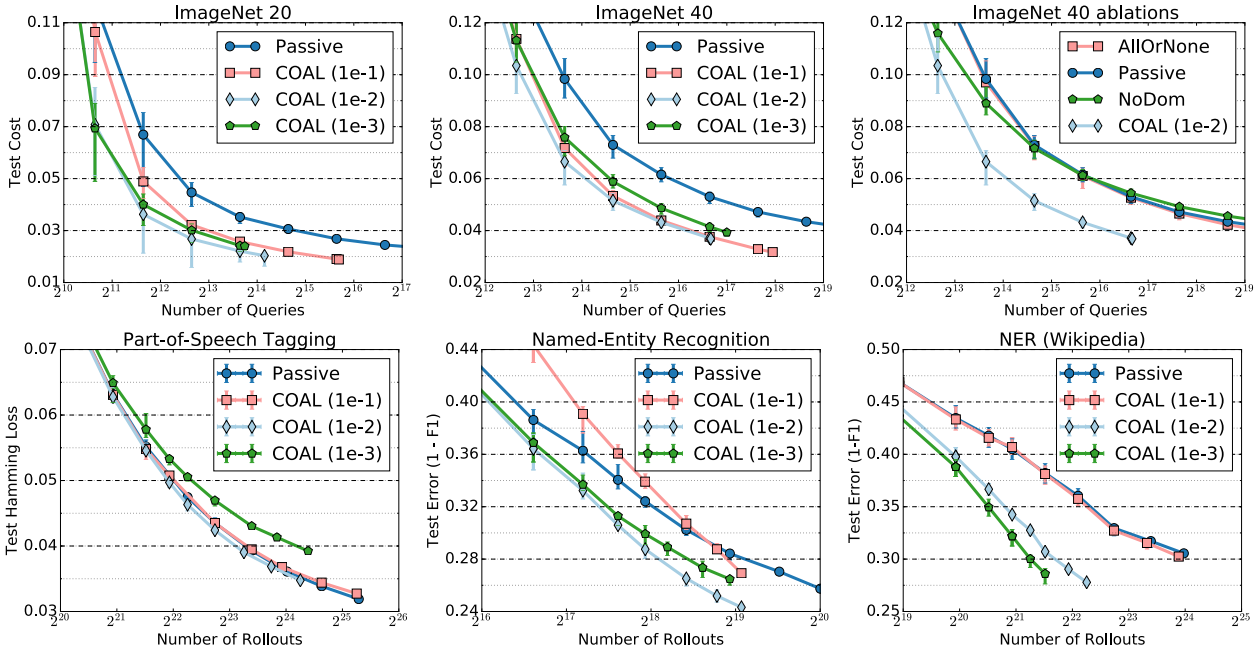

Figure 2. Experiments with COAL. Top row shows test cost vs. number of queries for simulated active learning experiments. Bottom row shows accuracy vs. number of rollouts for active and passive learning as the CSMC algorithm in learning-to-search.

where a sequence of decisions are made to generate the structured label. We focus here on sequence labeling tasks, where the input is a sentence and the output is a sequence of labels (specifically, parts of speech or named entities).

Learning-to-search solves joint prediction problems by generating the output one label at a time, conditioning the input x on all past decisions. Since mistakes may lead to compounding errors, it is natural to represent the decision space as a CSMC problem, where the classes are the "actions" available (possible labels for a word) and the costs reflect the long term loss of each choice. Intuitively, we should be able to avoid expensive computation of long term loss on decisions like "is 'the' a DETERMINER?" once we are quite sure of the answer. Similar ideas motivate adaptive sampling for structured prediction. (Shi et al., 2015).

We specifically use AGGRAVATE (Ross & Bagnell, 2014; Chang et al., 2015), which runs a learned policy to produce a backbone sequence of labels. For each position in the input, it then considers all possible deviation actions and executes an oracle for the rest of the sequence. The loss on this complete output is used as the cost for the deviating action. Run in this way, AGGRAVATE requires ℓK roll-outs when the input sentence has ℓ words and each word can take one of K possible labels.

Since each roll-out takes $\mathcal{O}(\ell)$ time, this can be computationally prohibitive, so we use active learning to reduce the number of roll-outs. We use COAL and a passive learning baseline inside AGGRAVATE on three joint prediction datasets (statistics are in Figure 2, upper right). As above, we use several mellowness values and the same AUC criteria to select the best learning rate. The results are in Figure 2, and again our recommended mellowness is 0.01.

Overall, active learning reduces the number of roll-outs required, but the improvements vary on the three datasets. On the Wikipedia data, COAL performs a factor of 4 less roll-outs to achieve similar performance to passive learning and achieves substantially better test performance. A similar, but less dramatic, behavior arises on the NER task. On the other hand, COAL offers minimal improvement over passive learning on the POS-tagging task. This agrees with our theory and prior empirical results (Hsu, 2010), which show that active learning may not always improve upon passive.

7. Discussion

This paper presents a new active learning algorithm for cost-sensitive multiclass classification. The algorithm enjoys strong theoretical guarantees and also outperforms passive baselines both in CSMC and structured prediction.

We close with some intriguing questions:

1. Can we use a square loss oracle in other partial information problems like contextual bandits?
2. Can we avoid the safety parameter to achieve the optimal complexity in the low noise case?
3. Can we analyze the online approximation to COAL?

We hope to answer these questions in future work.

References

Agarwal, A. Selective sampling algorithms for cost-sensitive multiclass prediction. In *International Conference on Machine Learning*, 2013.

Agarwal, A., Hsu, D., Kale, S., Langford, J., Li, L., and Schapire, R.E. Taming the monster: A fast and simple algorithm for contextual bandits. In *International Conference on Machine Learning*, 2014.

Balcan, M.-F. and Long, P. Active and passive learning of linear separators under log-concave distributions. In *Conference on Learning Theory*, 2013.

Balcan, M.-F., Beygelzimer, A., and Langford, J. Agnostic active learning. In *International Conference on Machine Learning*, 2006.

Balcan, M.-F., Broder, A., and Zhang, T. Margin based active learning. In *Conference on Learning Theory*, 2007.

Beygelzimer, A., Dasgupta, S., and Langford, J. Importance weighted active learning. In *International Conference on Machine Learning*, 2009.

Beygelzimer, A., Hsu, D., Langford, J., and Zhang, T. Agnostic active learning without constraints. In *Advances in Neural Information Processing Systems*, 2010.

Beygelzimer, A., Langford, J., Li, L., Reyzin, L., and Schapire, R.E. Contextual bandit algorithms with supervised learning guarantees. In *Artificial Intelligence and Statistics*, 2011.

Castro, R., Willett, R., and Nowak, R.D. Faster rates in regression via active learning. In *Advances in Neural Information Processing Systems*, 2005.

Castro, R.M. and Nowak, R.D. Minimax bounds for active learning. *Transaction on Information Theory*, 2008.

Cavallanti, G., Cesa-Bianchi, N., and Gentile, C. Learning noisy linear classifiers via adaptive and selective sampling. *Machine Learning*, 2011.

Chang, K.-W., Krishnamurthy, A., Agarwal, A., Daumé III, H., and Langford, J. Learning to search better than your teacher. In *International Conference on Machine Learning*, 2015.

Dasgupta, S., Hsu, D., and Monteleoni, C. A general agnostic active learning algorithm. In *Advances in Neural Information Processing Systems*, 2007.

Daumé III, H., Langford, J., and Marcu, D. Search-based structured prediction. *Machine Learning*, 2009.

Dekel, O., Gentile, C., and Sridharan, K. Robust selective sampling from single and multiple teachers. In *Conference on Learning Theory*, 2010.

Duchi, J., Hazan, E., and Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. In *Conference on Learning Theory*, 2010.

Hanneke, S. Theory of disagreement-based active learning. *Foundations and Trends in Machine Learning*, 2014.

Hsu, D. *Algorithms for Active Learning*. PhD thesis, University of California at San Diego, 2010.

Huang, T.-K., Agarwal, A., Hsu, D., Langford, J., and Schapire, R.E. Efficient and parsimonious agnostic active learning. In *Advances in Neural Information Processing Systems*, 2015.

Karampatziakis, N. and Langford, J. Online importance weight aware updates. In *Uncertainty in Artificial Intelligence*, 2011.

Langford, J. and Beygelzimer, A. Sensitive error correcting output codes. In *Conference on Learning Theory*, 2005.

Lewis, D.D., Yang, Y., Rose, T.G., and Li, F. Rcv1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 2004. Data available at http://www.jmlr.org/papers/volume5/lewis04a/lyrl2004_rcv1v2_README.htm.

Littlestone, N. and Warmuth, M.K. The weighted majority algorithm. In *Foundations of Computer Science*, 1989.

Massart, P. and Nédélec, É. Risk bounds for statistical learning. *The Annals of Statistics*, 2006.

Orabona, F. and Cesa-Bianchi, N. Better algorithms for selective sampling. In *International Conference on Machine Learning*, 2011.

Ross, S. and Bagnell, J.A. Reinforcement and imitation learning via interactive no-regret learning. *arXiv:1406.5979*, 2014.

Ross, S., Mineiro, P., and Langford, J. Normalized online learning. In *Uncertainty in Artificial Intelligence*, 2013.

Settles, B. Active learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 2012.

Shi, T., Steinhardt, J., and Liang, P. Learning where to sample in structured prediction. In *Artificial Intelligence and Statistics*, 2015.

Silla Jr., C.N. and Freitas, A.A. A survey of hierarchical classification across different application domains. *Data Mining and Knowledge Discovery*, 2011.

Syrgkanis, V., Krishnamurthy, A., and Schapire, R.E. Efficient algorithms for adversarial contextual learning. In *International Conference on Machine Learning*, 2016.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S.E., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. Going deeper with convolutions. In *Computer Vision and Pattern Recognition*, 2015.

Zhang, C. and Chaudhuri, K. Beyond disagreement-based agnostic active learning. In *Advances in Neural Information Processing Systems*, 2014.