

Two faces of active learning

Sanjoy Dasgupta
dasgupta@cs.ucsd.edu

Abstract

An active learner has a collection of data points, each with a label that is initially hidden but can be obtained at some cost. Without spending too much, it wishes to find a classifier that will accurately map points to labels. There are two common intuitions about how this learning process should be organized: (i) by choosing query points that shrink the space of candidate classifiers as rapidly as possible; and (ii) by exploiting natural clusters in the (unlabeled) data set. Recent research has yielded learning algorithms for both paradigms that are efficient, work with generic hypothesis classes, and have rigorously characterized labeling requirements. Here we survey these advances by focusing on two representative algorithms and discussing their mathematical properties and empirical performance.

1 Introduction

As digital storage gets cheaper, and sensing devices proliferate, and the web grows ever larger, it gets easier to amass vast quantities of unlabeled data – raw speech, images, text documents, and so on. But to build classifiers from these data, labels are needed, and obtaining them can be costly and time consuming. When building a speech recognizer for instance, the speech signal comes cheap but thereafter a human must examine the waveform and label the beginning and end of each phoneme within it. This is tedious and painstaking, and requires expertise.

We will consider situations in which we are given a large set of unlabeled points from some domain \mathcal{X} , each of which has a hidden label, from a finite set \mathcal{Y} , that can be queried. The idea is to find a good classifier, a mapping $h : \mathcal{X} \rightarrow \mathcal{Y}$ from a pre-specified set \mathcal{H} , without making too many queries. For instance, each $x \in \mathcal{X}$ might be the description of a molecule, with its label $y \in \{+1, -1\}$ denoting whether or not it binds to a particular target of interest. If the x 's are vectors, a possible choice of \mathcal{H} is the class of linear separators.

In this setting, a *supervised* learner would query a random subset of the unlabeled data and ignore the rest. A *semisupervised* learner would do the same, but would keep around the unlabeled points and use them to constrain the choice of classifier. Most ambitious of all, an *active learner* would try to get the most out of a limited budget by choosing its query points in an intelligent and adaptive manner (Figure 1).

1.1 A model for analyzing sampling strategies

The practical necessity of active learning has resulted in a glut of different querying strategies over the past decade. These differ in detail, but very often conform to the following basic paradigm:

Start with a pool of unlabeled data $S \subset \mathcal{X}$
Pick a few points from S at random and get their labels
Repeat:
 Fit a classifier $h \in \mathcal{H}$ to the labels seen so far
 Query the unlabeled point in S closest to the boundary of h
 (or most uncertain, or most likely to decrease overall uncertainty,...)

This high level scheme (Figure 2) has a ready and intuitive appeal. But how can it be analyzed?

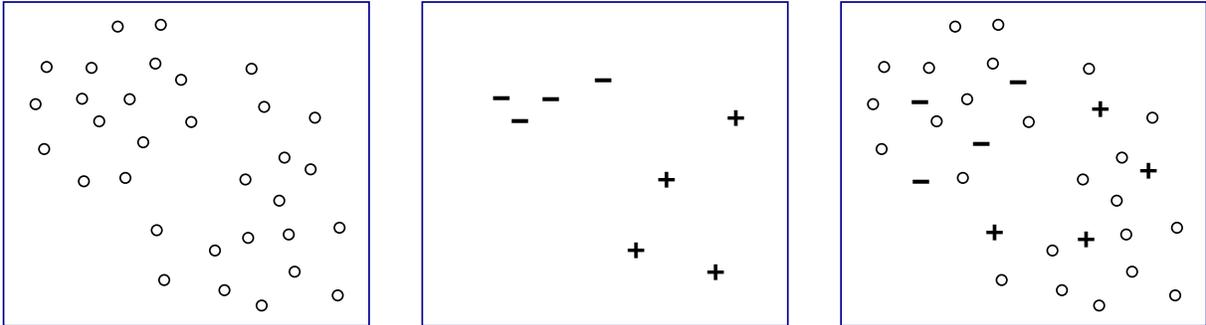


Figure 1: Each circle represents an unlabeled point, while + and - denote points of known label. *Left:* raw and cheap - a large reservoir of unlabeled data. *Middle:* supervised learning picks a few points to label and ignores the rest. *Right:* Semisupervised and active learning get more use out of the unlabeled pool, by using them to constrain the choice of classifier, or by choosing informative points to label.

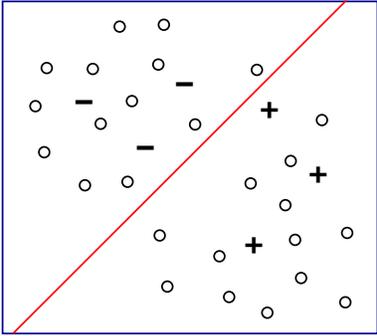


Figure 2: A typical active learning strategy chooses the next query point near the decision boundary obtained from the current set of labeled points. Here the boundary is a linear separator, and there are several unlabeled points close to it that would be candidates for querying.

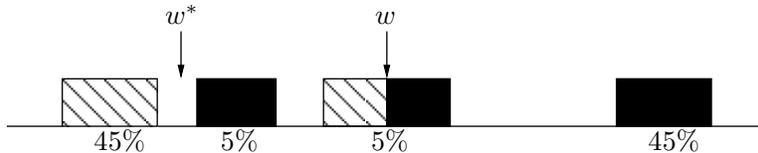


Figure 3: An illustration of sampling bias in active learning. The data lie in four groups on the line, and are (say) distributed uniformly within each group. The two extremal groups contain 90% of the distribution. Solids have a + label, while stripes have a - label.

To do so, we shall consider learning problems within the framework of *statistical learning theory*. In this model, there is an unknown, underlying distribution \mathbb{P} from which data points (and their hidden labels) are drawn independently at random. If \mathcal{X} denotes the space of data and \mathcal{Y} the labels, this \mathbb{P} is a distribution over $\mathcal{X} \times \mathcal{Y}$. Any classifier we build is evaluated in terms of its performance on \mathbb{P} .

In a typical learning problem, we choose classifiers from a set of candidate hypotheses \mathcal{H} . The best such candidate, $h^* \in \mathcal{H}$, is by definition the one with smallest error on \mathbb{P} , that is, with smallest

$$\text{err}(h) = \mathbb{P}[h(X) \neq Y].$$

Since \mathbb{P} is unknown, we cannot perform this minimization ourselves. However, if we have access to a sample of n points from \mathbb{P} , we can choose a classifier h_n that does well on this sample. We hope, then, that $h_n \rightarrow h^*$ as n grows. If this is true, we can also talk about the rate of convergence of $\text{err}(h_n)$ to $\text{err}(h^*)$.

A special case of interest is when h^* makes no mistakes: that is, $h^*(x) = y$ for all (x, y) in the support of \mathbb{P} . We will call this the *separable* case and will frequently use it in preliminary discussions because it is especially amenable to analysis. All the algorithms we describe here, however, are designed for the more realistic *nonseparable* scenario.

1.2 Sampling bias

When we consider the earlier querying scheme within the framework of statistical learning theory, we immediately run into the special difficulty of active learning: *sampling bias*.

The initial set of random samples from S , with its labeling, is a good reflection of the underlying data distribution \mathbb{P} . But as training proceeds, and points are queried based on increasingly confident assessments of their informativeness, the training set looks less and less like \mathbb{P} . It consists of an unusual subset of points, hardly a representative subsample; why should a classifier trained on these strange points do well on the overall distribution?

To make this intuition concrete, let's consider the simple one-dimensional data set depicted in Figure 3. Most of the data lies in the two extremal groups, so an initial random sample has a good chance of coming entirely from these. Suppose the hypothesis class consists of thresholds on the line: $\mathcal{H} = \{h_w : w \in \mathbb{R}\}$ where

$$h_w(x) = \begin{cases} +1 & \text{if } x \geq w \\ -1 & \text{if } x < w \end{cases} \quad \begin{array}{c} - \\ \text{-----} \\ + \\ | \\ w \end{array}$$

Then the initial boundary will lie somewhere in the center group, and the first query point will lie in this group. So will every subsequent query point, forever. As active learning proceeds, the algorithm will gradually converge to the classifier shown as w . But this has 5% error, whereas classifier w^* has only 2.5% error. Thus the learner is not consistent: even with infinitely many labels, it returns a suboptimal classifier.

The problem is that the second group from the left gets overlooked. It is not part of the initial random sample, and later on, the learner is mistakenly confident that the entire group has a - label. And this is just in one dimension; in high dimension, the problem can be expected to be worse, since there are more places

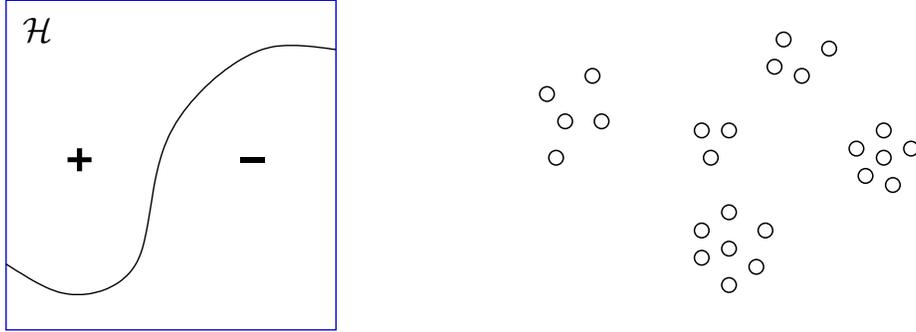


Figure 4: Two faces of active learning. *Left*: In the case of binary labels, each data point x cuts the hypothesis space \mathcal{H} into two pieces: the hypotheses that label it $+$, and those that label it $-$. If data is separable, one of these two pieces can be discarded once the label of x is known. A series of well-chosen query points could rapidly shrink \mathcal{H} . *Right*: If the unlabeled points look like this, perhaps we just need five labels.

for this troublesome group to be hiding out. For a discussion of this problem in text classification, see the paper of Schütze et al. [17].

Sampling bias is the most fundamental challenge posed by active learning. In this paper, we will deal exclusively with learning strategies that are provably consistent and we will analyze their *label complexity*: the number of labels queried in order to achieve a given rate of accuracy.

1.3 Two faces of active learning

Assuming sampling bias is correctly handled, how exactly is active learning helpful? The recent literature offers two distinct narratives for explaining this. The first has to do with *efficient search through the hypothesis space*. Each time a new label is seen, the current version space – the set of classifiers that are still “in the running” given the labels seen so far – shrinks somewhat. It is reasonable to try to explicitly select points whose labels will shrink this version space as fast as possible (Figure 4, left). Most theoretical work in active learning attempts to formalize this intuition. There are now several learning algorithms which, on a variety of canonical examples, provably yield significantly lower label complexity than supervised learning [9, 13, 10, 2, 3, 7, 15, 16, 12]. We will use one particular such scheme [12] as an illustration of the general theory.

The second argument for active learning has to do with *exploiting cluster structure in data*. Suppose, for instance, that the unlabeled points form five nice clusters (Figure 4, right); with luck, these clusters will be pure in their class labels and only five labels will be necessary! Of course, this is hopelessly optimistic. In general, there may be no nice clusters, or there may be viable clusterings at many different resolutions. The clusters themselves may only be mostly-pure, or they may not be aligned with labels at all. An ideal scheme would do something sensible for any data distribution, but would especially be able to detect and exploit any cluster structure (loosely) aligned with class labels, at whatever resolution. This type of active learning is a bit of an unexplored wilderness as far as theory goes. We will describe one preliminary piece of work [11] that solves some of the problems, clarifies some of the difficulties, and opens some doors to further research.

2 Efficient search through hypothesis space

The canonical example here is when the data lie on the real line and the hypotheses \mathcal{H} are thresholds. How many labels are needed to find a hypothesis $h \in \mathcal{H}$ whose error on the underlying distribution \mathbb{P} is at most some ϵ ?

	Separable data	General (nonseparable) data
Aggressive	Query by committee [13] Splitting index [10]	
Mellow	Generic mellow learner [9]	A^2 algorithm [2] Disagreement coefficient [15] Reduction to supervised [12] Importance-weighted approach [5]

Figure 5: Some of the key results on active learning within the framework of statistical learning theory. The *splitting index* and *disagreement coefficient* are parameters of a learning problem that control the label complexity of active learning. The other entries of the table are all learning algorithms.

In supervised learning, such issues are well understood. The standard machinery of sample complexity [6] tells us that if the data are *separable*—that is, if they can be perfectly classified by some hypothesis in \mathcal{H} —then we need approximately $1/\epsilon$ random labeled examples from \mathbb{P} , and it is enough to return any classifier consistent with them.

Now suppose we instead draw $1/\epsilon$ *unlabeled* samples from \mathbb{P} :



If we lay these points down on the line, their hidden labels are a sequence of $-$'s followed by a sequence of $+$'s, and the goal is to discover the point w at which the transition occurs. This can be accomplished with a binary search which asks for just $\log 1/\epsilon$ labels: first ask for the label of the median point; if it's $+$, move to the 25th percentile point, otherwise move to the 75th percentile point; and so on. Thus, for this hypothesis class, active learning gives an *exponential* improvement in the number of labels needed, from $1/\epsilon$ to just $\log 1/\epsilon$. For instance, if supervised learning requires a million labels, active learning requires just $\log 1,000,000 \approx 20$, literally!

This toy example is only for separable data, but with a little care something similar can be achieved for the nonseparable case. It is a tantalizing possibility that even for more complicated hypothesis classes \mathcal{H} , a sort of generalized binary search is possible.

2.1 Some results of active learning theory

There is a large body of work on active learning within the *membership query* model [1] and also some work on active *online learning* [8]. Here we focus exclusively on the framework of statistical learning theory, as described in the introduction. The results obtained so far can be categorized according to whether or not they are able to handle nonseparable data distributions, and whether their querying strategies are aggressive or mellow. This last dichotomy is imprecise, but roughly, an aggressive scheme is one that seeks out highly informative query points while a mellow scheme queries any point that is at all informative, in the sense that its label cannot be inferred from the data already seen.

Some representative results are shown in Figure 5. At present, the only schemes known to work in the realistic case of nonseparable data are mellow. It might seem that such schemes would confer at best a modest advantage over supervised learning — a constant-factor reduction in label complexity, perhaps. The surprise is that in a wide range of cases, they offer an exponential reduction.

2.2 A generic mellow learner

Cohn, Atlas, and Ladner [9] introduced a wonderfully simple, mellow learning strategy for separable data. This scheme, henceforth nicknamed CAL, has formed the basis for much subsequent work. It operates in a streaming model where unlabeled data points arrive one at a time, and for each, the learner has to decide on the spot whether or not to ask for its label.

$\mathcal{H}_1 = \mathcal{H}$ For $t = 1, 2, \dots$: Receive unlabeled point x_t If disagreement in \mathcal{H}_t about x_t 's label: query label y_t of x_t $\mathcal{H}_{t+1} = \{h \in \mathcal{H}_t : h(x_t) = y_t\}$ else: $\mathcal{H}_{t+1} = \mathcal{H}_t$	$S = \{\}$ (points seen so far) For $t = 1, 2, \dots$: Receive unlabeled point x_t If $\mathbf{learn}(S \cup (x_t, +1))$ and $\mathbf{learn}(S \cup (x_t, -1))$ both return an answer: query label y_t else: set y_t to whichever label succeeded $S = S \cup \{(x_t, y_t)\}$
---	--

Figure 6: *Left*: CAL, a generic mellow learner for separable data. *Right*: A way to simulate CAL without having to explicitly maintain the version space \mathcal{H}_t . Here $\mathbf{learn}(\cdot)$ is a black-box supervised learner that takes as input a data set and returns any classifier from \mathcal{H} consistent with the data, provided one exists.

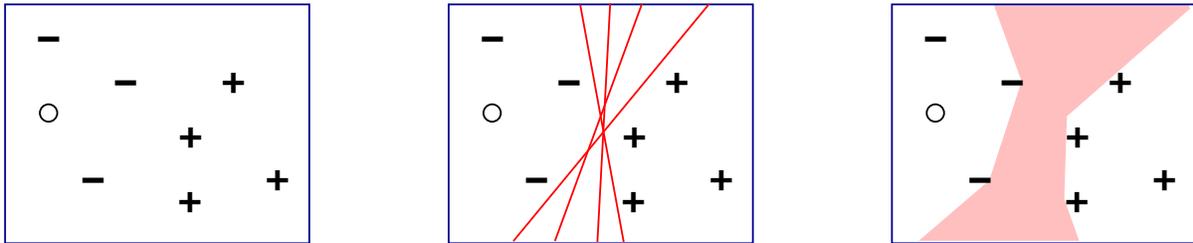


Figure 7: *Left*: The first seven points in the data stream were labeled. How about this next point? *Middle*: Some of the hypotheses in the current version space. *Right*: The region of disagreement.

CAL works by always maintaining the current version space: the subset of hypotheses consistent with the labels seen so far. At time t , this is some $\mathcal{H}_t \subset \mathcal{H}$. When the data point x_t arrives, CAL checks to see whether there is any disagreement within \mathcal{H}_t about its label. If there isn't, then the label can be inferred; otherwise it must be requested (Figure 6, left).

Figure 7 shows CAL at work in a setting where the data points lie in the plane, and the hypotheses are linear separators. A key concept is that of the *disagreement region*, the portion of the input space \mathcal{X} on which there is disagreement within \mathcal{H}_t . A data point is queried if and only if it lies in this region, and therefore the efficacy of CAL depends upon the rate at which the \mathbb{P} -mass of this region shrinks. As we will see shortly, there is a broad class of situations in which this shrinkage is geometric: the \mathbb{P} -mass halves every constant number of labels, giving a label complexity that is exponentially better than that of supervised learning. It is quite surprising that so mellow a scheme performs this well; and it is of interest, then, to ask how it might be made more practical.

2.3 Upgrading CAL

As described, CAL has two major shortcomings. First, it needs to explicitly maintain the version space, which is unmanageably large in most cases of interest. Second, it makes sense only for separable data. Two recent papers, first [2] and then [12], have shown how to overcome these hurdles.

The first problem is easily handled. The version space can be maintained implicitly in terms of the labeled examples seen so far (Figure 6, right). To handle the second problem, nonseparable data, we need to change the definition of the version space \mathcal{H}_t , because there may not be any hypotheses that agree with all the labels. Thereafter, with the newly-defined \mathcal{H}_t , we continue as before, requesting the label y_t of any point x_t for which there is disagreement within \mathcal{H}_t . If there is no disagreement, we “infer” the label of x_t – call this \hat{y}_t – and include it in the training set. Because of nonseparability, the inferred \hat{y}_t might differ

from the actual hidden label y_t . Regardless, every point gets labeled, one way or the other. The resulting algorithm, which we will call DHM after its authors [12], is presented in the appendix. Here we give some rough intuition.

After t time steps, there are t labeled points (some queried, some inferred). Let $\text{err}_t(h)$ denote the empirical error of h on these points, that is, the fraction of these t points that h gets wrong. Writing h_t for the minimizer of $\text{err}_t(\cdot)$, define

$$\mathcal{H}_{t+1} = \{h \in \mathcal{H}_t : \text{err}_t(h) \leq \text{err}_t(h_t) + \Delta_t\},$$

where Δ_t comes out of some standard generalization bound (DHM doesn't do this exactly, but is similar in spirit). Then the following assertions hold:

- The optimal hypothesis h^* (with minimum error on the underlying distribution \mathbb{P}) lies in \mathcal{H}_t for all t .
- Any inferred label is consistent with h^* (although it might disagree with the actual, hidden label).

Because all points get labeled, there is no bias introduced into the marginal distribution on \mathcal{X} . It might seem, however, that there is some bias in the conditional distribution of y given x , because the inferred labels can differ from the actual labels. The saving grace is that this bias shifts the empirical error of every hypothesis in \mathcal{H}_t by the same amount – because all these hypotheses agree with the inferred label – and thus the relative ordering of hypotheses is preserved.

In a typical trial of DHM (or CAL), the querying eventually concentrates near the decision boundary of the optimal hypothesis. In what respect, then, do these methods differ from the heuristics we described in the introduction? To understand this, let's return to the example of Figure 3. The first few data points drawn from this distribution may well lie in the far-left and far-right clusters. So if the learner were to choose a single hypothesis, it would lie somewhere near the middle of the line. But DHM doesn't do this. Instead, it maintains the entire version space (implicitly), and this version space includes all thresholds between the two extremal clusters. Therefore the second cluster from the left, which tripped up naive schemes, will not be overlooked.

To summarize, DHM avoids the consistency problems of many other active learning heuristics by (i) making confidence judgements based on the current version space, rather than the single best current hypothesis, and (ii) labeling all points, either by query or inference, to avoid skewing the distribution on \mathcal{X} .

2.4 Label complexity

The label complexity of supervised learning is quite well characterized by a single parameter of the hypothesis class called the *VC dimension* [6]: if this dimension is d , then a classifier whose error is within ϵ of optimal can be learned using roughly d/ϵ^2 labeled examples. In the active setting, further information is needed in order to assess label complexity [10]. For mellow active learning, Hanneke [15] identified a key parameter of the learning problem (hypothesis class as well as data distribution) called the *disagreement coefficient* and gave bounds for CAL in terms of this quantity. It proved similarly useful for analyzing DHM [12]. We will shortly define this parameter and see examples of it, but in the meanwhile, we take a look at the bounds.

Suppose data is generated independently at random from an underlying distribution \mathbb{P} . How many labels does CAL or DHM need before it finds a hypothesis with error less than ϵ , with probability at least $1 - \delta$? To be precise, for a specific distribution \mathbb{P} and hypothesis class \mathcal{H} , we define the label complexity of CAL, denoted $\mathcal{L}_{\text{CAL}}(\epsilon, \delta)$, to be the smallest integer t_o such that for all $t \geq t_o$,

$$\mathbb{P}[\text{some } h \in \mathcal{H}_t \text{ has } \text{err}(h) > \epsilon] \leq \delta.$$

In the case of DHM, the distribution \mathbb{P} might not be separable, in which case we need to take into account the best achievable error:

$$\nu = \inf_{h \in \mathcal{H}} \text{err}(h).$$

$\mathcal{L}_{\text{DHM}}(\epsilon, \delta)$ is then the smallest t_o such that

$$\mathbb{P}[\text{some } h \in \mathcal{H}_t \text{ has } \text{err}(h) > \nu + \epsilon] \leq \delta$$

for all $t \geq t_o$. In typical supervised learning bounds, and here as well, the dependence of $\mathcal{L}(\epsilon, \delta)$ upon δ is modest, at most $\text{poly} \log(1/\delta)$. To avoid clutter, we will henceforth ignore δ and speak only of $\mathcal{L}(\epsilon)$.

Theorem 1 [16] *Suppose \mathcal{H} has finite VC dimension d , and the learning problem is separable, with disagreement coefficient θ . Then*

$$\mathcal{L}_{\text{CAL}}(\epsilon) \leq \tilde{O}\left(\theta d \log \frac{1}{\epsilon}\right),$$

where the \tilde{O} notation suppresses terms logarithmic in d , θ , and $\log 1/\epsilon$.

A supervised learner would need $\Omega(d/\epsilon)$ examples to achieve this guarantee, so active learning yields an exponential improvement when θ is finite: its label requirement scales as $\log 1/\epsilon$ rather than $1/\epsilon$. And this is without any effort at finding maximally informative points!

In the nonseparable case, the label complexity also depends on the minimum achievable error within the hypothesis class.

Theorem 2 [12] *With parameters as defined above,*

$$\mathcal{L}_{\text{DHM}}(\epsilon) \leq \tilde{O}\left(\theta \left(d \log^2 \frac{1}{\epsilon} + \frac{d\nu^2}{\epsilon^2}\right)\right)$$

where $\nu = \inf_{h \in \mathcal{H}} \text{err}(h)$.

In this same setting, a supervised learner would require $\Omega((d/\epsilon) + (d\nu/\epsilon^2))$ samples. If ν is small relative to ϵ , we again see an exponential improvement from active learning; otherwise, the improvement is by the constant factor ν .

The second term in the label complexity is inevitable for nonseparable data.

Theorem 3 [5] *Pick any hypothesis class with finite VC dimension d . Then there exists a distribution \mathbb{P} over $\mathcal{X} \times \mathcal{Y}$ for which any active learner must incur a label complexity*

$$\mathcal{L}(\epsilon, 1/2) \geq \Omega\left(\frac{d\nu^2}{\epsilon^2}\right),$$

where $\nu = \inf_{h \in \mathcal{H}} \text{err}(h)$.

The corresponding lower bound for supervised learning is $d\nu/\epsilon^2$.

2.5 The disagreement coefficient

We now define the leading constant in both label complexity upper bounds: the *disagreement coefficient*.

To start with, the data distribution \mathbb{P} induces a natural metric on the hypothesis class \mathcal{H} : the distance between any two hypotheses is simply the \mathbb{P} -mass of points on which they disagree. Formally, for any $h, h' \in \mathcal{H}$, we define

$$d(h, h') = \mathbb{P}[h(X) \neq h'(X)],$$

and correspondingly, the closed ball of radius r around h is

$$B(h, r) = \{h' \in \mathcal{H} : d(h, h') \leq r\}.$$

Now, suppose we are running either CAL or DHM, and that the current version space is some $V \subset \mathcal{H}$. Then the only points that will be queried are those that lie within the disagreement region

$$\text{DIS}(V) = \{x \in \mathcal{X} : \text{there exist } h, h' \in V \text{ with } h(x) \neq h'(x)\}.$$

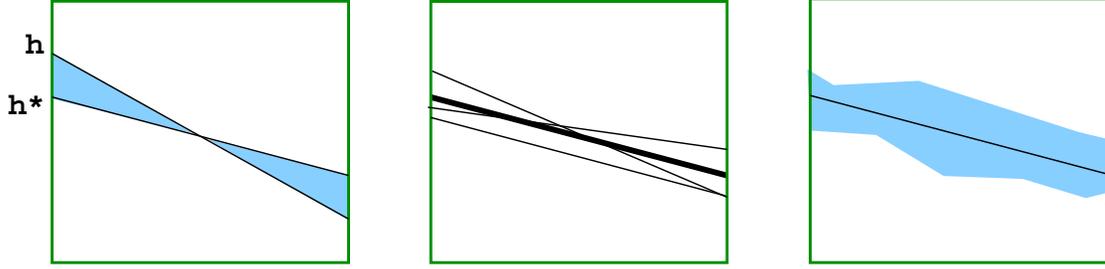


Figure 8: *Left*: Suppose the data lie in the plane, and that hypothesis class consists of linear separators. The distance between any two hypotheses h^* and h is the probability mass (under \mathbb{P}) of the region on which they disagree. *Middle*: The thick line is h^* . The thinner lines are examples of hypotheses in $B(h^*, r)$. *Right*: $\text{DIS}(B(h^*, r))$ might look something like this.

Figure 8 illustrates these notions.

If the minimum-error hypothesis is h^* , then after a certain amount of querying we would hope that the version space is contained within $B(h^*, r)$ for small-ish r . In which case, the probability that a random point from \mathbb{P} would get queried is at most $\mathbb{P}(\text{DIS}(B(h^*, r)))$. The disagreement coefficient measures how this probability scales with r : it is defined to be

$$\theta = \sup_{r>0} \frac{\mathbb{P}[\text{DIS}(B(h^*, r))]}{r}.$$

Let's work through an example. Suppose $\mathcal{X} = \mathbb{R}$ and \mathcal{H} consists of thresholds. For any two thresholds $h < h'$, the distance $d(h, h')$ is simply the \mathbb{P} -mass of the interval $[h, h')$. If h^* is the best threshold, then $B(h^*, r)$ consists exactly of the interval I that contains: h^* ; the segment to the immediate left of h^* of \mathbb{P} -mass r ; and the segment to the immediate right of h^* of \mathbb{P} -mass r . The disagreement region $\text{DIS}(B(h^*, r))$ is this same interval I ; and since it has mass $2r$, it follows that the disagreement coefficient is 2.

Disagreement coefficients have been derived for various concept classes and data distributions of interest, including:

- Thresholds in \mathbb{R} : $\theta = 2$, as explained above.
- Homogeneous (through-the-origin) linear separators in \mathbb{R}^d , with a data distribution \mathbb{P} that is uniform over the surface of the unit sphere [15]: $\theta \leq \sqrt{d}$.
- Linear separators in \mathbb{R}^d , with a smooth data density bounded away from zero [14]: $\theta = c(h^*)d$, where $c(h^*)$ is some constant depending on the target hypothesis h^* .

2.6 Further work on mellow active learning

A more refined disagreement coefficient

When the disagreement coefficient θ is bounded, CAL and DHM offer better label complexity than supervised learning. But there are simple instances in which θ is arbitrarily large. For instance, suppose again that $\mathcal{X} = \mathbb{R}$ but that the hypotheses consist of *intervals*:

$$\mathcal{H} = \{h_{a,b} : a, b \in \mathbb{R}\}, \quad h_{a,b}(x) = \begin{cases} +1 & \text{if } a \leq x \leq b \\ -1 & \text{otherwise} \end{cases}$$

Then the distance between any two hypotheses $h_{a,b}$ and $h_{a',b'}$ is

$$d(h_{a,b}, h_{a',b'}) = \mathbb{P}\{x : x \in [a, b] \cup [a', b'], x \notin [a, b] \cap [a', b']\} = \mathbb{P}([a, b] \Delta [a', b']),$$

where $S\Delta T$ denotes the symmetric set difference $(S \cup T) \setminus (S \cap T)$. Now suppose the target hypothesis is some $h_{\alpha,\beta}$ with $\alpha \leq \beta$. If $r > \mathbb{P}[\alpha, \beta]$ then $B(h_{\alpha,\beta}, r)$ includes all intervals of probability mass $\leq r - \mathbb{P}[\alpha, \beta]$. Thus, if \mathbb{P} is a density, the disagreement region of $B(h_{\alpha,\beta}, r)$ is all of \mathcal{X} ! Letting r approach $\mathbb{P}[\alpha, \beta]$ from above, we see that θ is at least $1/\mathbb{P}[\alpha, \beta]$, which is unbounded as β gets closer to α .

A saving grace is that for smaller values $r \leq \mathbb{P}[\alpha, \beta]$, the hypotheses in $B(h_{\alpha,\beta}, r)$ are intervals intersecting $h_{\alpha,\beta}$, and consequently the disagreement region has mass at most $4r$. Thus there are two regimes in the active learning process for \mathcal{H} : an initial phase in which the radius of uncertainty r is brought down to $\mathbb{P}[\alpha, \beta]$, and a subsequent phase in which r is further decreased to $O(\epsilon)$. The first phase might be slow, but the second should behave as if $\theta = 4$. Moreover, the dependence of the label complexity upon ϵ should arise entirely from the second phase. A series of recent papers [4, 14] analyzes such cases by loosening the definition of disagreement coefficient from

$$\sup_{r>0} \frac{\mathbb{P}[\text{DIS}(B(h^*, r))]}{r} \quad \text{to} \quad \limsup_{r \rightarrow 0} \frac{\mathbb{P}[\text{DIS}(B(h^*, r))]}{r}.$$

In the example above, the revised disagreement coefficient is 4.

Other loss functions

DHM uses a supervised learner as a black box, and assumes that this subroutine truly returns a hypothesis minimizing the empirical error. However, in many cases of interest, such as high-dimensional linear separators, this minimization is NP-hard and is typically solved in practice by substituting a convex loss function in place of $0 - 1$ loss. Some recent work [5] develops a mellow active learning scheme for general loss functions, using importance weighting.

2.7 Illustrative experiments

We now show DHM at work in a few toy cases. The first is our recurring example of threshold functions on the line. Figure 9 shows the results of an experiment in which the (unlabeled) data is distributed uniformly over the interval $[0, 1]$ and the target threshold is 0.5. Three types of noise distribution are considered:

- No noise: every point above 0.5 gets a label of +1 and the rest get a label of -1.
- Random noise: each data point's label is flipped with a certain probability, 10% in one experiment and 20% in another. This is a benign form of noise.
- Boundary noise: the noisy labels are concentrated near the boundary (for more details, see [12]). The fraction of corrupted labels is 10% in one experiment and 20% in another. This kind of noise is challenging for active learning, because the learner's region of disagreement gets progressively more noisy as it shrinks.

Each experiment has a stream of 10,000 unlabeled points; the figure shows how many of these are queried during the course of learning.

Figures 10 and 11 show similar results for the hypothesis class of intervals. The distribution of queries is initially all over the place, but eventually concentrates at the decision boundary of the target hypothesis. Although the behavior of this active learner seems qualitatively similar to that of the heuristics we described in the introduction, it differs in one crucial respect: here, *all* points get labeled to avoid bias in the training set.

The experiments so far are for one-dimensional data. There are two significant hurdles in scaling up the DHM algorithm to real data sets:

1. The version space \mathcal{H}_t is defined using a generalization bound. Current bounds are tight only in a few special cases, such as small finite hypothesis classes and thresholds on the line. Otherwise they can be extremely loose, with the result that \mathcal{H}_t ends up being larger than necessary, and far too many points get queried.

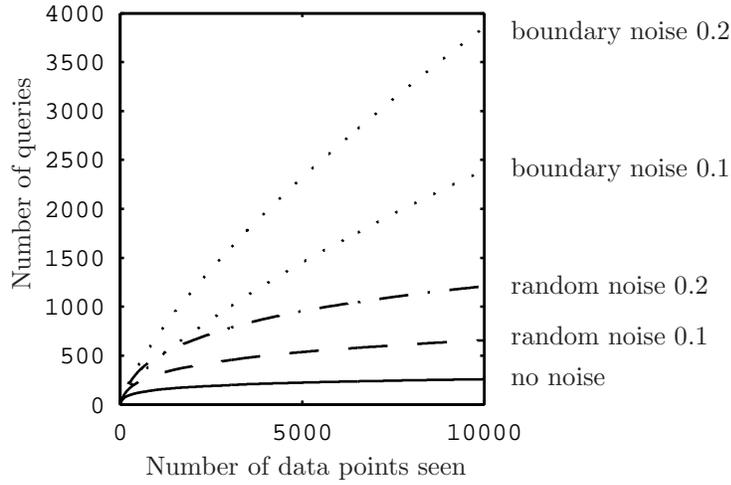


Figure 9: Here the data distribution is uniform over $\mathcal{X} = [0, 1]$ and \mathcal{H} consists of thresholds on the line. The target threshold is at 0.5. We test five different noise models for the conditional distribution of labels.

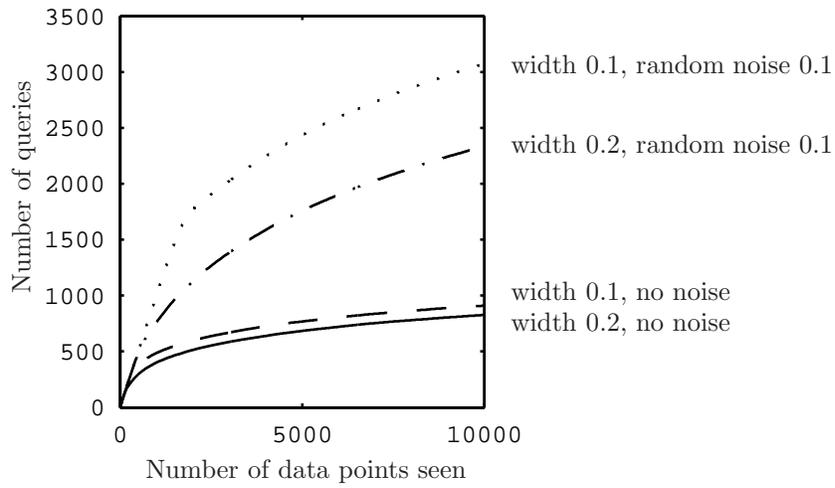


Figure 10: The data distribution is uniform over $\mathcal{X} = [0, 1]$, and \mathcal{H} consists of intervals. We vary the width of the target interval and the noise model for the conditional distribution of labels.

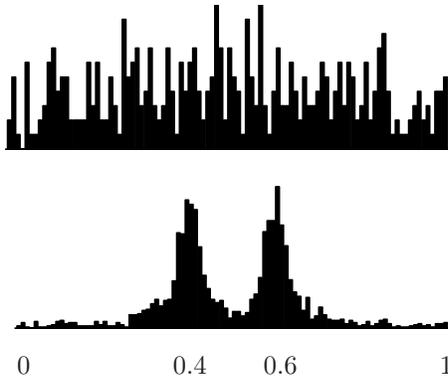


Figure 11: The distribution of queries for the experiment of Figure 10, with target interval $[0.4, 0.6]$ and random noise of 0.1. The initial distribution of queries is shown above and the eventual distribution below.

2. Its querying policy is not at all aggressive.

The first problem is perhaps eroding with time, as better and better bounds are found. How well would CAL/DHM perform if this problem vanished altogether? That is, what are the limits of mellow active learning?

One way to investigate this question is to run DHM with the best possible bound. Recall the definition of the version space:

$$\mathcal{H}_{t+1} = \{h \in \mathcal{H}_t : \text{err}_t(h) \leq \text{err}_t(h_t) + \Delta_t\},$$

where $\text{err}_t(\cdot)$ is the empirical error on the first t points, and h_t is the minimizer of $\text{err}_t(\cdot)$. Instead of obtaining Δ_t from large deviation theory, we can simply set it to the smallest value that retains the target hypothesis h^* within the version space; that is, $\Delta_t = \text{err}_t(h^*) - \text{err}_t(h_t)$. This cannot be used in practice because $\text{err}_t(h^*)$ is unknown; here we use it in a synthetic example to explore how effective mellow active learning could be if the problem of loose generalization bounds were to vanish.

Figure 12 shows the result of applying this optimistic bound to a ten-dimensional learning problem with two Gaussian classes. The hypotheses are linear separators and the best separator has an error of 5%. A stream of 500 data points is processed, and as before, the label complexity is impressive: less than 70 of these points get queried. The figure on the right presents a different view of the same experiment. It shows how the test error decreases over the first 50 queries of the active learner, as compared to a supervised learner (which asks for every point's label). There is an initial phase where active learning offers no advantage; this is the time during which a reasonably good classifier, with about 8% error, is learned. Thereafter, active learning kicks in and after 50 queries reaches a level of error that the supervised learner will not attain until after about 500 queries. However, this final error rate is still 5%, and so the benefit of active learning is realized only during the last 3% decrease in error rate.

2.8 Where next

The single biggest open problem is to develop active learning algorithms with more aggressive querying strategies. This would appear to significantly complicate problems of sampling bias, and has thus proved tricky to analyze mathematically. Such schemes might be able to exploit their querying ability earlier in the learning process, rather than having to wait until a reasonably good classifier has already been obtained.

Next, we'll turn to an entirely different view of active learning and present a scheme that is, in fact, able to realize early benefits.

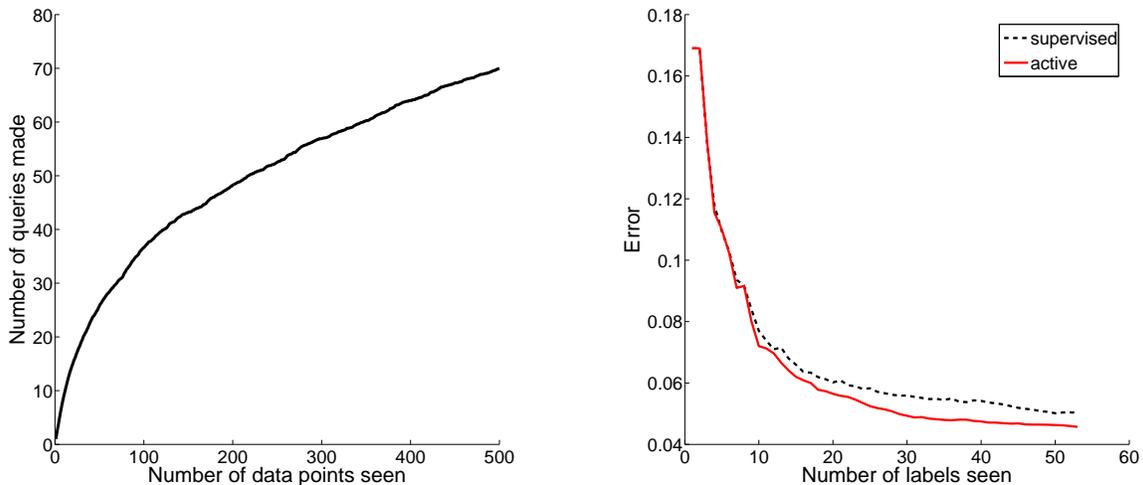


Figure 12: Here $\mathcal{X} = \mathbb{R}^{10}$ and \mathcal{H} consists of linear separators. Each class is Gaussian and the best separator has 5% error. *Left:* Queries made over a stream of 500 examples. *Right:* Test error for active versus supervised sampling.

3 Exploiting cluster structure in data

Figure 4, right, immediately brings to mind an active learning strategy that starts with a pool of unlabeled data, clusters it, asks for one label per cluster, and then takes these labels as representative of their entire respective clusters. Such a scheme is fraught with problems: (i) there might not be an obvious clustering, or alternatively, (ii) good clusterings might exist at many granularities (for instance, there might be a good partition into ten clusters but also into twenty clusters), or, worst of all, (iii) the labels themselves might not be aligned with the clusters.

What is needed is a scheme that does not make assumptions about the distribution of data and labels, but is able to exploit situations where there exist clusters that are fairly homogeneous in their labels. An elegant example of this is due to Zhu, Lafferty, and Ghahramani [20]. Their algorithm begins by imposing a neighborhood graph on an unlabeled data set S , and then works by locally propagating any labels it obtains, roughly as follows:

Pick a few points from S at random and get their labels
 Repeat:
 Propagate labels to “nearby” unlabeled points in the graph
 Query in an “unknown” part of the graph

This kind of nonparametric learner appears to have the usual problems of sampling bias, but differs from the approaches of the previous section, and has been studied far less. One recently-proposed algorithm, which we will call DH after its authors [11], attempts to capture the spirit of local propagation schemes while maintaining sound statistics on just how “unknown” different regions are; we now turn to it.

3.1 Three rules for querying

In the previous section the unlabeled points arrived in a *stream*, one at a time. Here we’ll assume we have all of them at the outset: a *pool* $S \subset \mathcal{X}$. The entire learning process will then consist of requesting labels of points from S .

In the DH algorithm, the only guide in deciding where to query is cluster structure. But the clusters in use may change over time. Let \mathbb{C}_t be the clustering operative at time t , while the t^{th} query is being chosen. This \mathbb{C}_t is some partition of S into groups; more formally,

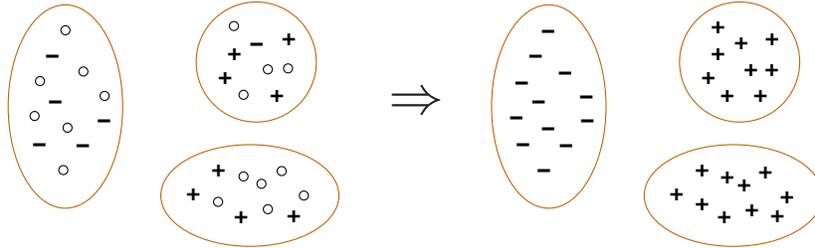
$$\bigcup_{C \in \mathbb{C}_t} C = S, \quad \text{and any } C, C' \in \mathbb{C}_t \text{ are either identical or disjoint.}$$

To avoid biases induced by sampling, DH follows a simple rule for querying:

Rule 1. At any time t , the learner specifies a cluster $C \in \mathbb{C}_t$ and receives the label of a point chosen uniformly at random from C (along with the identity of that point).

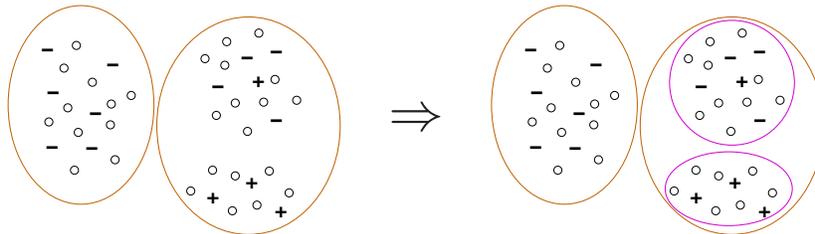
In other words, the learner is not allowed to specify exactly which point to label, but merely the cluster from which it will be randomly drawn. In the case where there are binary labels $\{-1, +1\}$, it helps to think of each cluster as a biased coin whose heads probability is simply the fraction of points within it with label $+1$. The querying process is then a toss of this biased coin. The scheme also works if the set of labels \mathcal{Y} is some finite set, in which case samples from the cluster are modeled by a multinomial.

If the labeling budget runs out at time T , the current clustering \mathbb{C}_T is used to assign labels to *all* of S : each point gets the majority label of its cluster.



To control the error induced by this process, it is important to find clusters that are as homogeneous as possible in their labels. In the above example, we can be fairly sure of this. We have five random labels from the left cluster, all of which are $-$. Using a tail bound for the binomial distribution, we can obtain an interval (such as $[0.8, 1.0]$) in which the true bias of this cluster is very likely to lie. The DH algorithm makes heavy use of such confidence intervals.

If the current clustering \mathbb{C} has a cluster that is very mixed in its labels, then this cluster needs to be split further, to get a new clustering \mathbb{C}' :



The lefthand cluster of \mathbb{C} can be left alone (for the time being), since the one on the right is clearly more troublesome. A fortunate consequence of Rule 1 is that the queries made to \mathbb{C} can be reused for \mathbb{C}' : a random label in the righthand cluster of \mathbb{C} is also a random label for the new cluster in which it falls.

Thus the clustering of the data changes only by splitting clusters.

Rule 2. Pick any two times $t' > t$. Then $\mathbb{C}_{t'}$ must be a *refinement* of \mathbb{C}_t , that is,

$$\text{for all } C' \in \mathbb{C}_{t'}, \text{ there exists some } C \in \mathbb{C}_t \text{ such that } C' \subset C.$$

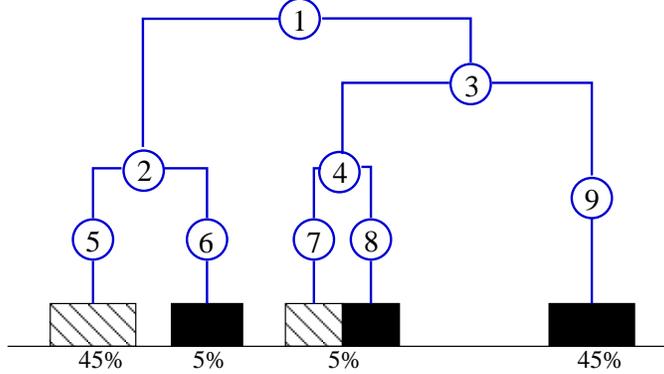


Figure 13: The top few nodes of a hierarchical clustering.

As in the example above, the nested structure of clusters makes it possible to re-use labels when the clustering changes. If at time t , the querying process yields (x, y) for some $x \in C \in \mathbb{C}_t$, then later, at $t' > t$, this same (x, y) is reusable as a random draw from the $C' \in \mathbb{C}_{t'}$ to which x belongs.

The final rule imposes a constraint on the manner in which a clustering is refined.

Rule 3. When a cluster is split to obtain a new clustering, $\mathbb{C}_t \rightarrow \mathbb{C}_{t+1}$, the manner of split cannot depend upon the labels seen.

This avoids complicated dependencies. The upshot of it is that we might as well start off with a hierarchical clustering of S , set \mathbb{C}_1 to the root of the clustering, and gradually move down the hierarchy, as needed, during the querying process.

3.2 An illustrative example

Given the three rules for querying, the specification of the DH cluster-based learner is quite intuitive. Let's get a sense of its behavior by revisiting the example in Figure 3 that presented difficulties for many active learning heuristics.

DH would start with a hierarchical clustering of this data. Figure 13 shows how it might look: only the top few nodes of the hierarchy are depicted, and their numbering is arbitrary. At any given time, the learner works with a particular partition of the data set, given by a pruning of the tree. Initially, this is just $\{1\}$, a single cluster containing everything. Random points are drawn from this cluster and their labels are queried. Suppose one of these points, x , lies in the rightmost group. Then it is a random sample from node 1 of the hierarchy, but also from nodes 3 and 9. Based on such random samples, each node of the tree maintains counts of the positive and negative instances seen within it. A few samples reveal that the top node 1 is very mixed while nodes 2 and 3 are substantially more pure. Once this transpires, the partition $\{1\}$ is replaced by $\{2, 3\}$. Subsequent random samples are chosen from either 2 or 3, according to a sampling strategy favoring the less-pure node. A few more queries down the line, the pruning is refined to $\{2, 4, 9\}$. This is when the benefits of the partitioning scheme become most obvious; based on the samples seen, it is concluded that cluster 9 is (almost) pure, and thus (almost) no more queries are made from it until the rest of the space has been partitioned into regions that are similarly pure.

The querying can be stopped at any stage; then, each cluster in the current partition is assigned the majority label of the points queried from it. In this way, DH labels the *entire* data set, while trying to keep the number of induced erroneous labels to a minimum. If desired, the labels can be used for a subsequent round of supervised learning, with any learning algorithm and hypothesis class.

```

 $P \leftarrow \{\text{root}\}$  (current pruning of tree)
 $L(\text{root}) \leftarrow 1$  (arbitrary starting label for root)
For  $t = 1, 2, \dots$  (until the budget runs out):
  Repeat  $B$  times:
     $v \leftarrow \text{select}(P)$ 
    Pick a random point  $z$  from subtree  $T_v$ 
    Query  $z$ 's label
    Update counts for all nodes  $u$  on path from  $z$  to  $v$ 
  In a bottom-up pass of  $T$ , compute  $\text{bound}(u)$  for all nodes  $u \in T$ 
  For each (selected)  $v \in P$ :
    Let  $(P', L')$  be the pruning and labeling of  $T_v$  minimizing  $\text{bound}(v)$ 
     $P \leftarrow (P \setminus \{v\}) \cup P'$ 
     $L(v) \leftarrow L'(u)$  for all  $u \in P'$ 
  For each cluster  $v \in P$ :
    Assign each point in  $T_v$  the label  $L(v)$ 

```

Figure 14: The DH cluster-adaptive active learning algorithm.

3.3 The learning algorithm

Figure 14 shows the DH active learning algorithm. Its input consists of a hierarchical clustering T whose leaves are the n data points, as well as a batch size B . This latter quantity specifies how many queries will be asked per iteration: it is frequently more convenient to make small batches of queries at a time rather than single queries. At the end of any iteration, the algorithm can be made to stop and output a labeling of all the data.

The pseudocode needs some explaining. The subtree rooted at a node $u \in T$ is denoted T_u . At any given time, the algorithm works with a particular *pruning* $P \subset T$, a set of nodes such that the subtrees (clusters) $\{T_u : u \in P\}$ are disjoint and contain all the leaves. Each node u of the pruning is assigned a label $L(u)$, with the intention that if the algorithm were to be abruptly stopped, all of the data points in T_u would be given this label. The resulting mislabeling error within T_u (that is, the fraction of T_u whose true label isn't $L(u)$) can be upper-bounded using the samples seen so far; $\text{bound}(u)$ is a conservative such estimate. These bounds can be computed directly from Hoeffding's inequality, or better still, from the tails of the binomial or multinomial; details are in [11]. Computationally, what matters is that after each batch of queries, all the bounds can be updated in a linear-time bottom-up pass through the tree, at which point the pruning P might be refined.

Two things remain unspecified: the manner in which the hierarchical clustering is built and the procedure **select**, which picks a cluster to query. We discuss these next, but regardless of how these decisions are made, the DH algorithm has a basic statistical soundness: it always maintains valid estimates of the error induced by its current pruning, and it refines prunings to drive down the error. This leaves a lot of flexibility to explore different clustering and sampling strategies.

The select procedure

The **select**(P) procedure controls the selective sampling. There are many choices for how to do this:

1. Choose $v \in P$ with probability $\propto w_v$.

This is similar to random sampling.

2. Choose $v \in P$ with probability $\propto w_v \text{bound}(v)$.

This is an active learning rule that reduces sampling in regions of the space that have already been observed to be fairly pure in their labels.

3. For each subtree $(T_z, z \in P)$, find the observed majority label, and assign this label to all points in the subtree; fit a classifier h to this data; and choose $v \in P$ with probability $\propto \min\{|\{x \in T_v : h(x) = +1\}|, |\{x \in T_v : h(x) = -1\}|\}$.

This biases sampling towards regions close to the current decision boundary.

Innumerable variations of the third strategy are possible. Such schemes have traditionally suffered from consistency problems (recall Figure 3), for instance because entire regions of space are overconfidently overlooked. The DH framework relieves such concerns because there is always an accurate bound on the error induced by the current labeling.

Building a hierarchical clustering

DH works best when there is a pruning P of the tree such that $|P|$ is small and a significant fraction of its constituent clusters are almost-pure. There are several ways in which one might try to generate such a tree. The first option is simply to run a standard hierarchical clustering algorithm, such as Ward’s average linkage method. If domain knowledge is available in the form of a specialized distance function, this can be used for the clustering; for instance, if the data is believed to lie on a low-dimensional manifold, a distance function can be generated by running Dijkstra’s algorithm on a neighborhood graph, as in Isomap [18]. A third option is to use a small set of labeled data to guide the construction of the hierarchical clustering, by providing soft constraints.

Label complexity

A rudimentary label complexity result for this model is proved in [11]: if the provided hierarchical clustering contains a pruning P whose clusters are ϵ -pure in their labels, then the learner will find a labeling that is $O(\epsilon)$ -pure with $O(|P|d(P)/\epsilon)$ labels, where $d(P)$ is the maximum depth of a node in P .

3.4 An illustrative experiment

The MNIST data set¹ is a widely-used benchmark for the multiclass problem of handwritten digit recognition. It contains over 60,000 images of digits, each a vector in \mathbb{R}^{784} . We began by extracting 10,000 training images and 2,000 test images, divided equally between the digits.

The first step in the DH active learning scheme is to hierarchically cluster the training data, bearing in mind that the efficacy of the subsequent querying process depends heavily on how well the clusters align with true classes. Previous work has found that the use of specialized distance functions such as tangent distance or shape context dramatically improves nearest-neighbor classification performance on this data set; the most sensible thing to do would therefore be to cluster using one of these distance measures. Instead, we tried our luck with Euclidean distance and a standard hierarchical agglomerative clustering procedure called Ward’s algorithm [19]. (Briefly, this algorithm starts with each data point in its own cluster, and then repeatedly merges pairs of clusters until there is a single cluster containing all the data. The merger chosen at each point in time is that which occasions the smallest increase in k -means cost.)

It is useful to look closely at the resulting hierarchical clustering, a tree with 10,000 leaves. Since there are ten classes (digits), the best possible scenario would be one in which there were a pruning consisting of ten nodes (clusters), each pure in its label. The worst possible scenario would be if purity were achieved only at the leaves – a pruning with 10,000 nodes. Reality thankfully falls much closer to the former extreme. Figure 15, left, relates the size of a pruning to the induced mislabeling error, that is, the fraction of points misclassified when each cluster is assigned its most-frequent label. For instance, the entry (50, 0.12) means that there exists a pruning with 50 nodes such that the induced label error is just 12%. This isn’t too bad given the number of classes, and bodes well for active learning.

A good pruning exists relatively high in the tree; but does the querying process find it or something comparable? The analysis shows that it must, using a number of queries roughly proportional to the number

¹<http://yann.lecun.com/exdb/mnist/>

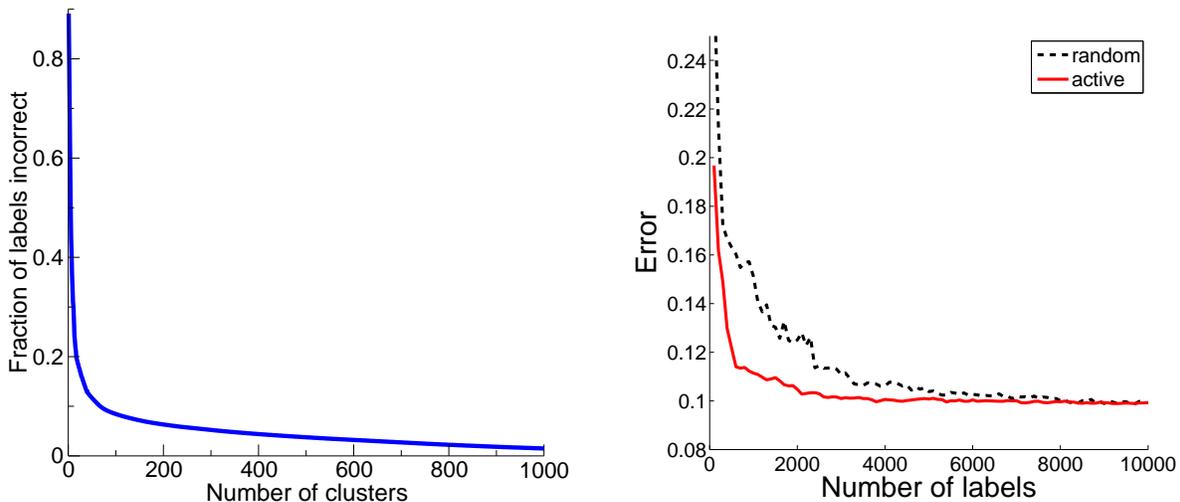


Figure 15: Results on OCR data. Left: Errors of the best prunings in the OCR digits tree. Right: Test error curves on classification task.

of clusters. As empirical corroboration, we ran the hierarchical sampler ten times, and on average 400 queries were needed to discover a pruning of error rate 12% or less.

So far we have only talked about error rates on the training set. We can complete the picture by using the final labeling from the sampling scheme as input to a supervised learner (logistic regression with ℓ_2 regularization, the trade-off parameter chosen by 10-fold cross validation). A good baseline for comparison is the same experiment but with random instead of active sampling. Figure 15, right, shows the resulting learning curves: the tradeoff between the number of labels and the error rate on the held-out test set. The initial advantage of cluster-adaptive sampling reflects its ability to discover and subsequently ignore relatively pure clusters at the onset of sampling. Later on, it is left sampling from clusters of easily confused digits (the prime culprits being 3’s, 5’s, and 8’s).

3.5 Where next

The DH algorithm attempts to capture the spirit of nonparametric local-propagation active learning, while maintaining reliable statistics about what it does and doesn’t know. In the process, however, it loses some flexibility in its sampling strategy – it would be useful to be able to relax this.

On the theory front, the label complexity of DH needs to be better understood. At present there is no clean characterization of when it does better than straight random sampling, and by how much. Among other things, this would help in choosing the querying rule (that is, the procedure `select`).

A very general way to describe this cluster-based active learner is to say it uses unlabeled data to create a data-dependent hypothesis class that is particularly amenable to adaptive sampling. This broad approach to active learning might be worth studying systematically.

Acknowledgements

The author is grateful for his collaborators – Alina Beygelzimer, Daniel Hsu, Adam Kalai, John Langford, and Claire Monteleoni – and for the support of the National Science Foundation under grant IIS-0713540. There were also two anonymous reviewers who gave very helpful feedback on the first draft of this paper.

$S = \emptyset$ (points with inferred labels)
 $T = \emptyset$ (points with queried labels)
 For $t = 1, 2, \dots$:
 Receive x_t
 If $(h_{+1} = \mathbf{learn}(S \cup \{(x_t, +1)\}, T))$ fails: Add $(x_t, -1)$ to S and break
 If $(h_{-1} = \mathbf{learn}(S \cup \{(x_t, -1)\}, T))$ fails: Add $(x_t, +1)$ to S and break
 If $\text{err}(h_{-1}, S \cup T) - \text{err}(h_{+1}, S \cup T) > \Delta_t$: Add $(x_t, +1)$ to S and break
 If $\text{err}(h_{+1}, S \cup T) - \text{err}(h_{-1}, S \cup T) > \Delta_t$: Add $(x_t, -1)$ to S and break
 Request y_t and add (x_t, y_t) to T

Figure 16: The DHM selective sampling algorithm. Here, $\text{err}(h, A) = (1/|A|) \sum_{(x,y) \in A} 1(h(x) \neq y)$. A possible setting for Δ_t is shown in Equation 1. At any time, the current hypothesis is $\mathbf{learn}(S, T)$.

4 Appendix: the DHM algorithm

For technical reasons, the DHM algorithm (Figure 16) makes black-box calls to a special type of supervised learner: for $A, B \subset \mathcal{X} \times \{\pm 1\}$,

$\mathbf{learn}(A, B)$ returns a hypothesis $h \in \mathcal{H}$ consistent with A , and with minimum error on B . If there is no hypothesis consistent with A , a failure flag is returned.

For some simple hypothesis classes like intervals on the line, or rectangles in \mathbb{R}^2 , it is easy to construct such a learner. For more complex classes like linear separators, the main bottleneck is the hardness of minimizing the 0 – 1 loss on B (that is, the hardness of agnostic supervised learning).

During the learning process, DHM labels every point but divides them into two groups: those whose labels are queried (denoted T in the figure) and those whose labels are inferred (denoted S). Points in S are subsequently used as hard constraints; with high probability, all these labels are consistent with the target hypothesis h^* .

The generalization bound Δ_t can be set to

$$\Delta_t = \beta_t^2 + \beta_t \left(\sqrt{\text{err}(h_{+1}, S \cup T)} + \sqrt{\text{err}(h_{-1}, S \cup T)} \right), \quad \beta_t = C \sqrt{\frac{d \log t + \log(1/\delta)}{t}} \quad (1)$$

where C is a universal constant, d is the VC dimension of class \mathcal{H} , and δ is the overall permissible failure probability.

References

- [1] D. Angluin. Queries revisited. In *Proceedings of the Twelfth International Conference on Algorithmic Learning Theory*, pages 12–31, 2001.
- [2] M.-F. Balcan, A. Beygelzimer, and J. Langford. Agnostic active learning. In *International Conference on Machine Learning*, 2006.
- [3] M.-F. Balcan, A. Broder, and T. Zhang. Margin based active learning. In *Conference on Learning Theory*, 2007.
- [4] M.-F. Balcan, S. Hanneke, and J. Wortman. The true sample complexity of active learning. In *Proceedings of the 21st Annual Conference on Learning Theory*, 2008.
- [5] A. Beygelzimer, S. Dasgupta, and J. Langford. Importance weighted active learning. In *International Conference on Machine Learning*, 2009.

- [6] O. Bousquet, S. Boucheron, and G. Lugosi. Introduction to statistical learning theory. *Lecture Notes in Artificial Intelligence*, 3176:169–207, 2004.
- [7] R. Castro and R. Nowak. Minimax bounds for active learning. *IEEE Transactions on Information Theory*, 54(5):2339–2353, 2008.
- [8] N. Cesa-Bianchi, C. Gentile, and L. Zaniboni. Worst-case analysis of selective sampling for linear-threshold algorithms. In *Advances in Neural Information Processing Systems*, 2004.
- [9] D. Cohn, L. Atlas, and R. Ladner. Improving generalization with active learning. *Machine Learning*, 15(2):201–221, 1994.
- [10] S. Dasgupta. Coarse sample complexity bounds for active learning. In *Neural Information Processing Systems*, 2005.
- [11] S. Dasgupta and D.J. Hsu. Hierarchical sampling for active learning. In *International Conference on Machine Learning*, 2008.
- [12] S. Dasgupta, D.J. Hsu, and C. Monteleoni. A general agnostic active learning algorithm. In *Neural Information Processing Systems*, 2007.
- [13] Y. Freund, H. Seung, E. Shamir, and N. Tishby. Selective sampling using the query by committee algorithm. *Machine Learning*, 28(2):133–168, 1997.
- [14] E. Friedman. Active learning for smooth problems. In *Conference on Learning Theory*, 2009.
- [15] S. Hanneke. A bound on the label complexity of agnostic active learning. In *International Conference on Machine Learning*, 2007.
- [16] S. Hanneke. *Theoretical Foundations of Active Learning*. PhD Thesis, CMU Machine Learning Department, 2009.
- [17] H. Schütze, E. Velipasaoglu, and J. Pedersen. Performance thresholding in practical text classification. In *ACM International Conference on Information and Knowledge Management*, 2006.
- [18] J. Tenenbaum, V. de Silva, and J. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.
- [19] J.H. Ward. Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, 58:236–244, 1963.
- [20] X. Zhu, J. Lafferty, and Z. Ghahramani. Combining active learning and semi-supervised learning using gaussian fields and harmonic functions. In *ICML Workshop on the Continuum from Labeled to Unlabeled Data*, 2003.