Algorithms for minimally supervised learning

Sanjoy Dasgupta
dasgupta@cs.ucsd.edu
UC San Diego

November 9, 2013

1 Introduction

The past few decades have brought substantial progress in the mathematical analysis of *supervised learning*. This is a paradigm in which a learner is provided with a data set consisting of points x and their labels (or response values) y, and is tasked with finding a suitable classifier (or regressor) that maps $x \to y$. There are many popular types of classifiers—decision trees, linear separators, boosted ensembles, and so on—and for almost all of these, there are by now fairly well-developed analyses of how they may be learned from data (the *computational* or *algorithmic* task of learning the function) and of how well they are likely to perform on future data (the *statistical* problem of generalization). Some good textbooks on this material include those of Mitchell (1997), Hastie et al. (2009), and Mohri et al. (2012).

This broad theoretical understanding quickly vanishes, however, as one moves away from supervised learning towards some other paradigms of statistics and machine learning. Take *clustering*, for example: the problem of partitioning a data set into groups. Like supervised learning, this task has a long history—recall the work of Pearson (1894) on clustering crabs using Gaussian mixture models—and has been studied just as intensively. Yet even the most basic computational and statistical questions around clustering are unresolved. A common way to define a clustering task is via a cost function, such as k-means or the log likelihood of a mixture model. This seems like it might bring clarity to the computational aspect of clustering, but in fact these optimization problems are typically NP-hard, meaning that they are computationally intractable to solve optimally. The heuristics used in practice are generally hill-climbing methods whose solutions can be arbitrarily far from optimal in the worst case, and there is little understanding of how well they perform on average, or on "well-behaved" data sets. The statistical picture is equally murky. Consistency results are absent for most clustering methods, and in the few instances where they do exist, such as for the k-means cost function, they apply only if the estimator is able to optimally solve its (intractable) optimization problem.

In these notes, we'll start by covering two recent strains of work on clustering that address the computational and statistical questions raised above. They are promising beginnings, that combine efficient algorithms with strong statistical guarantees of convergence, and they open the door to a more general theoretical understanding of clustering.

We will then move to another topic that has seen recent progress: exploiting the *intrinsic dimensionality* of data. High-dimensional spaces present many statistical challenges, most of all for nonparametric estimators, which in general require a number of samples exponential in the dimension in order to achieve a specified error rate. This would seem to entirely rule out nonparametric methods in today's world, where data sets are growing steadily larger and higher-dimensional. Yet there has been a resurgence of interest in these methods, based on the insight that a lot of data have fewer "degrees of freedom" than their dimension would suggest.

Perhaps the most popular manifestation of this idea is in the field of "manifold learning", which looks for ways of handling data that are arbitrarily high dimensional but lie on, or near, a low-dimensional manifold. There are many elegant heuristics that embed a data set of this type in a lower-dimensional space while attempting to preserve certain types of local geometry such as geodesic distances. This embedding can be

done in an entirely unsupervised manner (that is, requiring only the features x, not the labels/responses y), and the resulting representation can, for instance, later be used for supervised learning, presumably requiring more modest resources (measured by the number of labels y, or the computational complexity) than would have been needed in the original space. Although this is a compelling narrative, the consistency theory for these manifold learning heuristics is still primitive, and it is not clear that they are able to reliably capture the manifold structure of realistic data distributions using realistic sample sizes. In all, this embedding problem appears to be difficult.

In recent years, an intermediate option has proved fruitful: to give up on trying to precisely capture the low-dimensional structure and instead to merely construct a simple representation that coarsely exploits it. This project has led to fairly general notions of intrinsic dimension, spatial data structures that automatically adapt to them and are computationally efficient to construct, and nonparametric methods that use these structures to obtain better rates of convergence. These will be the second topic of these notes.

We will end with *active learning*, which is rather like supervised learning, but recognizes that in many situations, the points x are easy to obtain while their labels y come later, at a cost. For instance, in an image classification problem, it is easy to download millions of images from the internet. But labeling these requires human attention and is costly; thus, it will typically only be possible to label some small fraction of them. Which ones should be chosen?

The active learning formalization treats raw data points as cheap and their labels as expensive, and considers an adaptive querying process whose goal is to obtain a good classifier (that generalizes well to unseen data) at low cost. As with the other topics we consider, this is an eminently practical problem, with a variety of heuristics that are used in practice, but for which even basic issues of consistency are not resolved. Over the past decade, there has been a series of results clarifying some of the basic difficulties in active learning and providing efficient algorithms for which the tradeoff between the number of label queries, the number of unlabeled points, and the final classifier error rate can be cleanly characterized.

In all the problems we study, viable solutions must possess both computational and statistical guarantees—efficient algorithms and good rates of convergence—and these twin constraints are non-trivial to meet.

2 Clustering

The most popular clustering algorithm is probably the *k-means* heuristic (Forgy, 1965; Lloyd, 1982), which appears to first have been proposed by Lloyd: his 1982 paper appeared as a Bell Laboratories technical report in 1957. Given a set of points $S \subset \mathbb{R}^d$ and an integer k, it returns k "centers" $T = \{\mu_1, \ldots, \mu_k\} \subset \mathbb{R}^d$ with the goal of minimizing the average squared distance from a point to its closest center,

$$\text{cost}(T) = \sum_{x \in S} \min_{1 \le j \le k} \|x - \mu_j\|^2.$$

(Here, and in the rest of this article, $\|\cdot\|$ denotes Euclidean distance.) The algorithm works as follows:

1. Initialize centers $\mu_1, \ldots, \mu_k \in \mathbb{R}^d$ in any way
2. Repeat until there is no further change in cost:
   - For each j: let $C_j = \{x \in S \text{ whose closest center is } \mu_j\}$
   - For each j: let $\mu_j = \text{mean}(C_j)$

It is easy to see that an iteration of this procedure cannot increase the cost. Moreover, any set of k centers induces a corresponding partition $\{C_1, \ldots, C_k\}$ of S, and such partitions are finite in number. Hence the procedure converges in a finite number of steps to a local optimum of the cost function. Can anything more be said of it?

A deeper analysis must specify the method of initialization, which is known in practice to have a significant influence upon the quality of the final answer. But it can be shown that even under some popular and sensible-seeming initialization methods, the solution found by this heuristic can be arbitrarily far from the optimal

solution, in the sense of having a cost whose ratio to the optimal cost is arbitrarily large. We won't elaborate upon this here because we will shortly encounter a similar phenomenon in the case of the EM algorithm.

Given these difficulties, one might ask whether there is a different algorithm that is guaranteed to minimize the k-means cost function? The bad news is that this optimization problem is NP-hard, and thus highly unlikely to be efficient solvable, even if $k = 2$ (Dasgupta, 2008) or $d = 2$ (Vattani, 2009a; Mahajan et al., 2012); see Garey and Johnson (1979) for an overview of the theory of NP-completeness.

A common sentiment about Lloyd's k-means heuristic is that even if it isn't optimal, it is at least fast. Each iteration of the procedure takes just $O(k|S|)$ time, and it has often been noted experimentally that the number of iterations until convergence is small. However, it was recently shown by Vattani (2009b) that this is not true in general. He exhibited a data set consisting of n points in the plane, and a particular initialization, for which Lloyd's method takes $2^{\Omega(n)}$ iterations.

All in all, there is little joy to be had in a worst-case analysis of k-means. A similar situation prevails for many other common clustering procedures, including the EM (expectation-maximization) algorithm for mixture models and the various agglomerative hierarchical procedures.

What if we look at "well-behaved" instances? A natural way to do this is to consider a data set that has been generated i.i.d. from some underlying distribution P that in some sense matches the inherent assumptions of the clustering procedure. We now consider two such cases. The first is when P is a mixture of Gaussians, in which case the most commonly-used heuristic is EM, a close cousin of Lloyd's algorithm. We then turn to a nonparametric setting, in which P is arbitrary and the goal is to recover the *cluster tree* corresponding to it.

2.1 The EM algorithm for mixtures of spherical Gaussians

The canonical generative model for clustered data is the *mixture of Gaussians*; for information on mixture models see, for instance, the classic text of Titterington et al. (1985). A mixture of k Gaussians in $\mathbb{R}^d$ is specified by k mixing weights $w_1, \ldots, w_k > 0$ that sum to one, and by k Gaussian distributions $N(\mu_1, \Sigma_1), \ldots, N(\mu_k, \Sigma_k)$, where the μ_j are points in $\mathbb{R}^d$ and the Σ_j are $d \times d$ covariance matrices. A random sample X is generated from the mixture distribution as follows.

- Pick $J \in \{1, \ldots, k\}$ with $\Pr(J = j) = w_j$.
- Pick $X \sim N(\mu_J, \Sigma_J)$.

A special case of theoretical and practical interest is when the Gaussians are constrained to be *spherical*, that is, when each covariance matrix Σ_j is of the form $\sigma_j^2 I_d$ for some $\sigma_j > 0$ (and I_d is the $d \times d$ identity matrix). We will deal mostly with this case, and will return to the more general situation only in Section 2.3.

Given samples $S = \{x_1, \ldots, x_n\} \subset \mathbb{R}^d$ drawn i.i.d. from a mixture of k spherical Gaussians, how can these Gaussians (and the underlying clusters) be recovered? The standard method for doing this is the *expectation-maximization* (EM) procedure (Dempster et al., 1977; Wu, 1983; Redner and Walker, 1984). It begins by choosing starting values $\widehat{w}_j$, $\widehat{\mu}_j$, $\widehat{\sigma}_j$ for the parameters, and then updates them iteratively according to the two steps in Figure 1. How good a job does this algorithm do?

2.1.1 Deficiencies of an optimization-theoretic analysis

The tradition way of viewing EM is as an optimization procedure. Let $\Theta \subset \mathbb{R}^{k(d+2)}$ denote the space of possible parameter values: mixing weights, means, and variances. EM can be seen as hill-climbing the *log-likelihood* surface

$$\mathrm{LL}(\theta) = \sum_{i=1}^{n} \log P_\theta(x_i)$$

where P_θ denotes the mixture density corresponding to parameters $\theta \in \Theta$. Indeed, it is well known that this quantity does not decrease over iterations of EM.

Under these circumstances, the following basic questions need to be addressed:

E step Let $\tau_j(x)$ denote the density of the jth Gaussian-estimate, $N(\widehat{\mu}_j, \widehat{\sigma}_j^2 I_d)$. For each data point x_i and each $1 \leq j \leq k$, compute the conditional probability that x_i comes from the jth Gaussian with respect to the current estimated parameters:

$$p_j(x_i) = \frac{\widehat{w}_j \tau_j(x_i)}{\sum_{j'} \widehat{w}_{j'} \tau_{j'}(x_i)}.$$

M step Update the parameter-estimates:

$$\widehat{w}_j = \frac{1}{n}\sum_{i=1}^{n} p_j(x_i), \quad \widehat{\mu}_j = \frac{1}{n\widehat{w}_j}\sum_{i=1}^{n} p_j(x_i) x_i, \quad \widehat{\sigma}_j^2 = \frac{1}{n\widehat{w}_j d}\sum_{i=1}^{n} p_j(x_i)\|x_i - \widehat{\mu}_j\|^2$$

Figure 1: The EM algorithm alternates between these two steps until convergence.

1. If the data $x_1, \ldots, x_n$ are generated from a mixture model θ^*, is $\mathrm{LL}(\cdot)$ maximized near θ^*?
2. Does EM always find the global optimum of LL? If not, how close does it come?
3. How many iterations does EM require to converge (or to come close to convergence)?

The bad news begins with question 1, for which the answer is no. The maximum value of the log-likelihood is always infinity, and is achieved far from the generating mixture. To see this, suppose a finite data set is drawn from a mixture θ^* of two well-separated Gaussians. Now consider a different mixture θ that contains one Gaussian G_1 of high variance and another Gaussian G_2 that is centered at one of the data points (say x_1) and has miniscule variance. Under G_1, every data point has positive probability. Meanwhile, by letting the variance of G_2 shrink, the probability of point x_1 can be driven arbitrarily high. As this variance goes to zero, $\mathrm{LL}(\theta)$ goes to ∞.

Turning to questions 2 and 3, the optimization-theoretic view sheds little light on whether EM comes close to the generating mixture θ^*, or on how long it takes to converge. The standard form of analysis is to perform a Taylor expansion of the log-likelihood in the vicinity of a local optimum and thereby to get insight into the behavior of the algorithm when it is on the brink of convergence; but this says nothing about what happens prior to that moment.

An entirely different approach is to ignore the cost function that EM is ostensibly optimizing and to focus instead upon the specific *algorithm*: given data from a mixture model θ^*, what is the distribution of this algorithm's parameter-estimates on iteration 1, on iteration 2, and so on? This view turns out to be fruitful, as we will now see. The remainder of this section is taken from Dasgupta and Schulman (2007); all details can be found in that paper.

We begin by looking more closely at high-dimensional Gaussians.

2.1.2 Concentration properties of high-dimensional Gaussians

A spherical Gaussian $N(\mu, \sigma^2 I_d)$ assigns to point $x \in \mathbb{R}^d$ the density

$$p(x) = \frac{1}{(2\pi)^{d/2}\sigma^d} \exp\left(-\frac{\|x-\mu\|^2}{2\sigma^2}\right),$$

where $\|\cdot\|$ is Euclidean distance. For $X \sim N(\mu, \sigma^2 I_d)$, the individual coordinates of $X - \mu$ are independent with distribution $N(0, \sigma^2)$. Thus $\|X - \mu\|^2$ is a sum of independent random variables, with expected value $\sigma^2 d$. A large deviation bound is quite easily obtained for it.

Lemma 1 *Pick any $0 < \epsilon < 1$. For $X \sim N(\mu, \sigma^2 I_d)$,*

$$\Pr\left((1-\epsilon)\sigma^2 d \leq \|X-\mu\|^2 \leq (1+\epsilon)\sigma^2 d\right) \geq 1 - \exp(-c_o \epsilon^2 d),$$

where c_o is an absolute constant.

PROOF: The coordinates of $(X-\mu)/\sigma$ are i.i.d. standard normals. Thus $Y = \|X-\mu\|^2/\sigma^2$ is chi-squared with d degrees of freedom, and this distribution is well-known to have moment-generating function $\mathbb{E}[e^{tY}] = (1-2t)^{-d/2}$. By Markov's inequality, for $0 \le t \le 1/2$,

$$\Pr(Y \ge (1+\epsilon)d) \;=\; \Pr\left(\exp(tY) \ge \exp(t(1+\epsilon)d)\right) \;\le\; \frac{\mathbb{E}[e^{tY}]}{e^{t(1+\epsilon)d}} \;=\; \left(\frac{1}{(1-2t)e^{2t(1+\epsilon)}}\right)^{d/2}.$$

The upper bound on Y follows by taking $t = \epsilon/(2(1+\epsilon))$, and the lower bound can be obtained similarly, by working with $-Y$. □

The upshot of this lemma is that when the dimension d is high, almost the entire probability mass of $N(\mu, \sigma^2 I_d)$ lies in a thin shell at a radius of about $\sigma\sqrt{d}$ from μ.

Just as the distance of a point from its cluster center is tightly concentrated, so is the distance between two points in the same cluster, or in different clusters. To see why, let X and Y be draws from two, possibly identical, Gaussians $N(\mu_1, \sigma_1^2 I_d)$ and $N(\mu_2, \sigma_2^2 I_d)$. Then $X - Y$ also has a Gaussian distribution, $N(\mu_1 - \mu_2, (\sigma_1^2 + \sigma_2^2) I_d)$, and therefore we can obtain results like Lemma 1 for its squared norm. In particular, we can show that if a polynomial (in d) number of samples is drawn from a mixture of spherical Gaussians in $\mathbb{R}^d$, then with high probability, all interpoint distances will be close to their expected values.

Lemma 2 *There is an absolute constant $A > 0$ for which the following holds. Draw n points at random from a mixture of k spherical Gaussians. Let S_j denote the points from the jth Gaussian. Pick any $0 < \epsilon < 1$. Then, with probability at least $1 - kn^2 e^{-A\epsilon^2 d}$, the following hold simultaneously.*

(a) For any $x, y \in S_j$, we have $\|x-y\|^2 = 2\sigma_j^2 d(1 \pm \epsilon)$.

(b) For $x \in S_i, y \in S_j, i \ne j$, we have $\|x-y\|^2 = (\sigma_i^2 d + \sigma_j^2 d + \|\mu_i - \mu_j\|^2)(1 \pm 2\epsilon)$.

(c) For any $y \in S_j$, we have $\|y - \mu_j\|^2 = \sigma_j^2 d(1 \pm \epsilon)$ while for $i \ne j$, we have $\|y - \mu_i\|^2 = (\sigma_j^2 d + \|\mu_i - \mu_j\|^2)(1 \pm 2\epsilon)$.

These are favorable conditions: taking $\epsilon \sim d^{-1/2}$ in this lemma, we see that as long as there is a modest separation between the Gaussians—a distance of $\Omega(d^{1/4})$ between the means—it should be possible to cluster the points correctly by looking only at interpoint distances. Nonetheless, many clustering algorithms, including k-means, fail in these circumstances because they have trouble with clusters of varying radius. EM, on the other hand, holds promise.

Define the *separation* of a mixture $w_1 N(\mu_1, \sigma_1^2 I_d) + \cdots + w_k N(\mu_k, \sigma_k^2 I_d)$ as

$$s = \min_{i \ne j} \frac{\|\mu_i - \mu_j\|}{\max(\sigma_i, \sigma_j)}.$$

For instance, if the Gaussian means are at least δ standard deviations apart on each coordinate, then $s \ge \delta\sqrt{d}$. We will see that if $s \gg d^{1/4}$, a particular variant of EM will, with high probability, recover the generating mixture.

2.1.3 A variant of EM

Initialization

Let S denote the entire data set, with S_j being the points drawn from the jth Gaussian $N(\mu_j, \sigma_j^2 I_d)$. A common way to initialize EM for a mixture of k components is to pick k data points at random from S, and to use these as initial center-estimates $\widehat{\mu}_j$.

Here is a concrete example (Figure 2) of how this particular initialization can fail dramatically. In a suitably high dimension d, suppose the k Gaussians $N(\mu_1, I_d), \ldots, N(\mu_k, I_d)$ are side by side in a line, with a distance of at least $3\sqrt{d}$ between consecutive means, and with equal mixing weights. When initial centers are picked randomly from the data, there is a constant probability that they will include nothing from S_1,

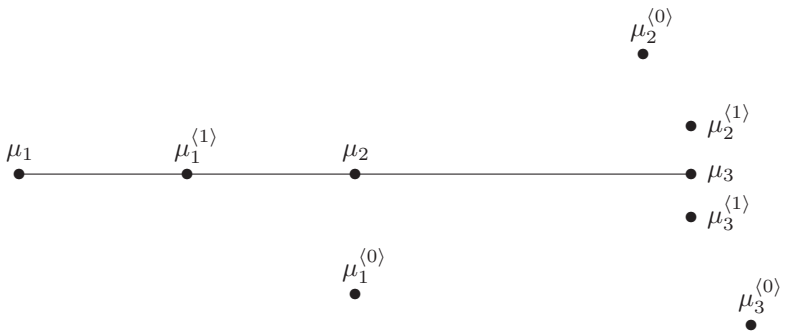


Figure 2: The μ_i are the true Gaussian means, while $\mu_i^{\langle t\rangle}$ are estimates are time t. For this mixture, the positions of the center-estimates do not move much after the first step of EM.

one point from S_2, and at least one point from S_3. Then no matter how long EM is run, it will assign just one Gaussian-estimate to the first two clusters. In the first round of EM, the point from S_2 (call it $\widehat{\mu}_1$) will move between μ_1 and μ_2. It will stay there, right between the two true centers. None of the other center-estimates $\widehat{\mu}_j$ will ever come closer to μ_2; their distance from it is too large to overwhelm the influence of $\widehat{\mu}_1$. This argument can be formalized easily using the concentration bounds above.

One way to avoid this bad situation is to ensure that the initial centers include at least one representative from each of the clusters S_i. This can be assured with high probability by starting with $\Omega(k \log k)$ initial centers, picked at random from the data. We also need a careful estimator of the initial variances: the distance concentration results suggest that

$$\widehat{\sigma}_j^2 = \frac{1}{2d} \min_{j \neq i} \|\widehat{\mu}_i - \widehat{\mu}_j\|^2$$

will be tightly concentrated around the variance of the Gaussian from which $\widehat{\mu}_i$ is drawn, provided there are at least two center-estimates from this cluster.

Having extra centers at the outset necessitates a later pruning step in which the number of centers is reduced to k. What about the behavior of EM iterations in the interim?

The first iteration of EM

Distance concentration effects guarantee that if the separation is at least $s \gg d^{1/4}$, the "soft assignments" $p_j(x)$ of the very first "E" step will assign almost the entire weight of a point $x \in S_j$ to center-estimates drawn from S_j.

Now let's consider what happens in the "M" step. There might be many initial center-estimates $\widehat{\mu}_i$ that were drawn from S_j. Each of them has two possible fates in the first round of EM:

1. The total weight of data assigned to it is very small. In this case, $\widehat{\mu}_i$ will get a small mixing weight $\widehat{w}_i$: a *starved cluster*.

2. A reasonable amount of data is assigned to $\widehat{\mu}_i$. As discussed above, the overwhelming majority of this data will be from S_j. Thus the mean computed in the "M" step will be quite close to the actual mean of the jth Gaussian.

The second conclusion warrants a little more discussion. Suppose the total weight of the data assigned to $\widehat{\mu}_i$ from S_j is t. How far can this weighted average of points in S_j lie from μ_j? It is not hard to see that (once t is rounded up to an integer) the worst case is when the weights are not fractional—each point in S_j either gets weight 0 or 1—whereupon the following bound can be applied.

1. **Initialization:** Pick ℓ data points at random as starting estimates $\widehat{\mu}_j$ for the Gaussian centers. Assign them identical mixing weights $\widehat{w}_j = 1/\ell$. For initial estimates of the variances use $\widehat{\sigma}_j^2 = (1/2d)\min_{j \neq i} \|\widehat{\mu}_i - \widehat{\mu}_j\|^2$.
2. **EM:** Run one round of EM. This yields modified estimates $\widehat{w}_j, \widehat{\mu}_j, \widehat{\sigma}_j$.
3. **Pruning:** Remove all center-estimates whose mixing weights are below $1/(4\ell)$. Then, prune the remaining center-estimates down to k using the following clustering procedure:
   - Compute distances between center-estimates: $d(\widehat{\mu}_i, \widehat{\mu}_j) = \|\widehat{\mu}_i - \widehat{\mu}_j\|/(\widehat{\sigma}_i + \widehat{\sigma}_j)$.
   - Choose one of these centers arbitrarily.
   - Pick the remaining $k-1$ iteratively as follows: pick the center farthest from the ones picked so far. (The distance from a point x to a set S is $\min_{y \in S} d(x, y)$.)

   Re-number the resulting center-estimates 1 to k and set all the mixing weights to $\widehat{w}_j = 1/k$.
4. **EM:** Run one more step of EM, yielding the final parameter estimates.

Figure 3: A two-round variant of EM. Here $\ell = \Omega(k \log k)$.

Lemma 3 *Pick a set of n points independently at random from $N(0, I_d)$. Choose any integer $1 \leq t \leq n$. Then with probability at least $1 - e^{-d/2}$, no subset of t or more of the points has mean of ℓ_2 norm greater than $\sqrt{6\max(d/t, \ln(ne/t))}$.*

PROOF: It is enough to show this for subsets of size exactly t. The mean of t points drawn independently from $N(0, I_d)$ has distribution $N(0, (1/t)I_d)$; we can therefore bound its norm using Lemma 1. We finish with a union bound over all $\binom{n}{t}$ possible subsets. □

Thus, running EM will produce center-estimates that are either close to true cluster means or correspond to starved clusters. Moreover, at least one of the center-estimates associated with any cluster S_j will be of the former variety.

Pruning the center-estimates

Since we start with more than k center-estimates, they need to be pruned. One way to do this is to first remove any starved clusters, and to then run a simple clustering heuristic on the remaining centers that picks exactly one center-estimate $\widehat{\mu}_i$ per true cluster. This requires some care in correctly handling the variability in cluster radii: see step 3 of Figure 3.

Algorithm and result

With exactly one center-estimate per (true) Gaussian, a second iteration of EM will accurately retrieve the means, variances, and mixing weights. In fact, due to the concentration effects of Lemma 1, the clustering of the data (the fractional labels assigned by EM) will be almost perfect, that is to say, each fractional label will be close to zero or one, and will in almost all cases correctly identify the generating Gaussian. The final algorithm, shown in Figure 3, admits the following guarantee.

Theorem 4 *Say n data points are generated from a mixture of k Gaussians in $\mathbb{R}^d$,*

$$w_1 N(\mu_1, \sigma_1^2 I_d) + \cdots + w_k N(\mu_k, \sigma_k^2 I_d),$$

where the intercenter distances satisfy the inequality $\|\mu_i - \mu_j\|^2 \geq |\sigma_i^2 - \sigma_j^2|d$.

Define s to be the separation, $\min_{i \neq j} \|\mu_i - \mu_j\|/\max(\sigma_i, \sigma_j)$. Let S_i denote the points from the ith Gaussian, and let $w_{\min} = \min_i w_i$. For any $0 < \delta, \epsilon < 1$, if

- *parameter* $\ell = \widetilde{\Omega}((1/w_{\min})\ln(1/w_{\min}))$
- *dimension* $d = \widetilde{\Omega}(\ln(1/w_{\min}))$
- *number of samples* $n = \Omega(\ell \max(1, d/s^2))$
- *separation* $s^2 = \widetilde{\Omega}(d^{1/2}\ln(1/w_{\min}))$

(where $\widetilde{\Omega}$ *conceals terms logarithmic in* d*,* $1/\epsilon$*, and* $1/\delta$*) then with probability at least* $1-\delta$*, the variant of EM described above will produce final center-estimates* $\widehat{\mu}_j$ *which (appropriately permuted) satisfy*

$$\|\widehat{\mu}_j - \mu_j\| \;\leq\; \|\text{mean}(S_j) - \mu_j\| + \epsilon\sigma_j\sqrt{d}.$$

Bounds can be also be given for the final mixing weights and variances.

In summary, if EM is initialized properly, it can be shown to do well on high-dimensional mixtures of spherical Gaussians for which the distances between Gaussians satisfy $\|\mu_i - \mu_j\| \gg \max(\sigma_i, \sigma_j)d^{1/4}$. It is an open problem to determine what happens at lower separations, or with non-spherical Gaussians.

2.2 Spectral methods for mixtures of spherical Gaussians

The EM procedure is interesting to study in detail because it is widely used in practice and, more technically, is a convenient setting for bringing out the concentration properties of Gaussian mixtures. Previous analyses have typically ignored the *algorithm* and focused instead upon its *cost function*; by doing the reverse, we obtained a recovery guarantee when the Gaussian centers in $\mathbb{R}^d$ are $\gg d^{1/4}$ apart.

Over the past decade, a variety of other algorithms have been developed with progressively milder separation assumptions. A key tool in achieving this has been the use of *singular value decomposition*.

2.2.1 Principal component analysis as a prelude to clustering

Let P denote an unknown mixture of spherical Gaussians, $w_1 N(\mu_1, \sigma_1^2 I_d) + \cdots + w_k N(\mu_k, \sigma_k^2 I_d)$. The second moment of P is a $d \times d$ matrix,

$$\mathbb{E}_{X\sim P}[XX^T] \;=\; \sum_{j=1}^{k} w_j\, \mathbb{E}[XX^T | X \sim N(\mu_j, \sigma_j^2 I_d)] \;=\; \sum_{j=1}^{k} w_j(\mu_j\mu_j^T + \sigma_j^2 I_d) \;=\; \sum_{j=1}^{k} w_j\mu_j\mu_j^T + \sigma^2 I_d, \quad (1)$$

where we define $\sigma^2 = \sum_j w_j\sigma_j^2$. This matrix can be estimated from samples and we henceforth assume for simplicity that we have a perfect estimate; the effects of sampling error can be quantified using matrix perturbation theory.

Suppose principal component analysis (PCA) is used and the data are projected to the subspace spanned by the top k eigenvectors of $\mathbb{E}[XX^T]$, assuming the common situation $d > k$. The projected distribution is also a mixture of spherical Gaussians, with the same mixing weights w_j and the same variances σ_j^2. A fundamental insight of Vempala and Wang (2004) is that distances between Gaussian means also remain unchanged. To state the result, let $\Pi : \mathbb{R}^d \to \mathbb{R}^k$ denote the PCA projection.

Theorem 5 *For all* $1 \leq i, j \leq k$*, we have* $\|\Pi(\mu_i) - \Pi(\mu_j)\| = \|\mu_i - \mu_j\|$.

PROOF: We can write $\Pi(x) = V^T x$, where V is the $d \times k$ matrix whose columns correspond to the top k eigenvectors of $\mathbb{E}[XX^T]$; in particular, they have unit length and are orthogonal (so $V^TV = I_k$). It is well known from the Rayleigh quotient characterization that V is the maximizer of $\mathbb{E}_{X\sim P}[\|V^TX\|^2]$ over all orthonormal transformations of the same dimension. Writing out this function and using $\|z\|^2 = \text{tr}(zz^T)$, we get

$$\mathbb{E}[\|V^TX\|^2] \;=\; \mathbb{E}[\text{tr}(V^TXX^TV)] \;=\; \text{tr}(V^T\mathbb{E}[XX^T]V).$$

We can then substitute the expression (1) above to get

$$\mathbb{E}[\|V^T X\|^2] \;=\; \mathrm{tr}\left(\sum_{j=1}^{k} w_j V^T \mu_j \mu_j^T V + \sigma^2 V^T V\right) \;=\; \sum_{j=1}^{k} w_j \|V^T \mu_j\|^2 + \sigma^2 k.$$

This is maximized when $\|V^T \mu_j\| = \|\mu_j\|$, that is, when the means μ_j lie in the column space of V—whereupon Π preserves the lengths of all vectors in $\mathrm{span}(\mu_1, \ldots, \mu_k)$. $\square$

Thus the projection does not change the "signal"—the intercluster distances $\|\mu_i - \mu_j\|$—but decreases the "noise"—the cluster radii—from $\sigma_j\sqrt{d}$ to $\sigma_j\sqrt{k}$. If we use EM after projection, the clusters will be correctly recovered when the original separation is $s \gg k^{1/4}$, a potentially much milder condition than $s \gg d^{1/4}$.

2.2.2 A third moment

We have seen that the second moment $\mathbb{E}[XX^T]$ can be exploited to reduce the separation requirements for learning a Gaussian mixture model. A remarkable result of Hsu and Kakade (2013) shows that it is possible to do away with separation requirements altogether by using a *third* moment. As before, we'll assume for convenience that we can perfectly compute low-order moments of the distribution.

First of all, notice from (1) that all but the k largest eigenvalues of $\mathbb{E}[XX^T]$ are equal to σ^2. Thus we can easily recover σ^2 as well as a corresponding eigenvector v_o. The following can then be obtained:

- The d-dimensional vector $M_1 = \mathbb{E}[X(v_o^T(X - \mathbb{E}X))^2]$.
- The $d \times d$ matrix $M_2 = \mathbb{E}[X \otimes X] - \sigma^2 I_d$.
- The $d \times d \times d$ tensor $M_3 = \mathbb{E}[X \otimes X \otimes X] - \sum_{i=1}^{d}(M_1 \otimes e_i \otimes e_i + e_i \otimes M_1 \otimes e_i + e_i \otimes e_i \otimes M_1)$.

It can be verified by algebraic manipulation that these are exactly:

$$M_1 \;=\; \sum_{i=1}^{k}(w_i\sigma_i^2)\mu_i, \quad M_2 \;=\; \sum_{i=1}^{k} w_i(\mu_i \otimes \mu_i), \quad M_3 \;=\; \sum_{i=1}^{k} w_i(\mu_i \otimes \mu_i \otimes \mu_i),$$

so that we have "denoised" versions of the second and third moments. These contain information about the means μ_i, but how can the means be extracted?

The tensor M_3 is unwieldy and is not needed in full. To get a two-dimensional projection, we take a vector $\eta \in \mathbb{R}^d$ and define $M_3(\eta)$ as the $d \times d$ matrix with entries

$$M_3(\eta)_{ij} = \sum_{\ell} M_{3,ij\ell}\, \eta_\ell.$$

This will be enough for parameter recovery.

A suggestive way to rewrite these various quantities is as matrix-vector products. Let Φ be the $d \times k$ matrix whose columns are the means μ_i and let $w = (w_1, \ldots, w_k)$. Then:

- $M_1 = \Phi\, \mathrm{diag}(\sigma_1^2, \ldots, \sigma_k^2)\, w$.
- $M_2 = \Phi\, \mathrm{diag}(w)\, \Phi^T$.
- $M_3(\eta) = \Phi\, \mathrm{diag}(w)\, \mathrm{diag}(\eta^T\mu_1, \ldots, \eta^T\mu_k)\, \Phi^T$.

It seems plausible that there is a way to combine the singular value decompositions of M_2 and $M_3(\eta)$ so as to retrieve Φ.

In particular, we take the following steps:

1. Compute the nonzero eigenvalue-eigenvector pairs $(\lambda_1, v_1), (\lambda_2, v_2), \ldots$ of $(M_2^+)^{1/2} M_3(\eta)(M_2^+)^{1/2}$, where M_2^+ is the Moore-Penrose pseudoinverse of M_2. It can be checked that this product matrix is of the form $V \mathrm{diag}(\eta^T\mu_1, \ldots, \eta^T\mu_k)V^T$, where $V^TV = I_k$. Hence $\lambda_i = \eta^T\mu_i$ for $1 \le i \le k$, upto permutation of the cluster labels. Moreover $M_2^{1/2} v_i = \pm w_i^{1/2} \mu_i$.

2. We can thus recover the means using

$$\widehat{\mu}_i = \frac{\lambda_i}{\eta^T M_2^{1/2} v_i} M_2^{1/2} v_i,$$

and, letting $\widehat{\Phi}$ be the $d \times k$ matrix whose columns are these $\widehat{\mu}_i$,

$$\widehat{w}_i = e_i^T \widehat{\Phi}^+ (\mathbb{E}X), \quad \widehat{\sigma}_i^2 = \frac{1}{\widehat{w}_i} e_i^T \widehat{\Phi}^+ M_1.$$

Theorem 6 *Let Φ denote the $d \times k$ matrix whose columns are the means μ_i. If Φ has column rank k, and w is strictly positive, and η is picked at random from the unit sphere in $\mathbb{R}^d$, then with probability 1, this procedure correctly recovers the parameters of the mixture model (up to permutation of the component labels).*

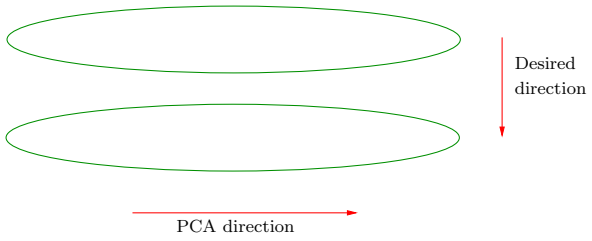
The authors (Hsu and Kakade, 2013) show that in the case where the moments are estimated from a finite sample drawn i.i.d. from the Gaussian mixture, the resulting error in the estimates can be brought below ϵ by choosing a number of samples polynomial in d, k, $1/\epsilon$, and terms involving the kth largest singular value of M_2. It is an open problem to determine whether a better rate of convergence can be achieved by a polynomial-time procedure.

The critical assumption on the mixture is that the matrix of means, Φ, has rank k. A lower bound of Kalai et al. (2012) suggests that this condition might be necessary for any estimator whose resources scale only polynomially in k, but a precise characterization has yet to be obtained.

2.3 Beyond spherical Gaussians

We have seen some elegant and practical algorithms for recovering a mixture of spherical Gaussians given data sampled from it. A more difficult case is when the individual covariances are allowed to be arbitrary positive definite matrices, resulting in clusters with ellipsoidal contours.

Consider, for instance, the simple observation of Theorem 5, that PCA is a sensible prelude to clustering when the clusters are spherical. In the ellipsoidal case, PCA might produce misleading directions: a canonical example is when there are two clusters with means μ_1 and μ_2 and with a common covariance matrix Σ that is highly elongated along a direction orthogonal to $\mu_1 - \mu_2$. If we wish to project on a single direction, the ideal choice for purposes of clustering is $\mu_1 - \mu_2$, but PCA will choose the top eigenvector of Σ, which is orthogonal to this (see picture below). A precise lower bound that captures this limitation of PCA was obtained by Achlioptas and McSherry (2005).



In terms of positive results, suppose we have an algorithm for spherical Gaussians that works when the mixture $\{N(\mu_i, \sigma_i^2 I_d)\}$ has an intercluster separation that is bounded below,

$$\min_{i \neq j} \frac{\|\mu_i - \mu_j\|}{\max(\sigma_i, \sigma_j)} \ \ge \ s_o,$$

for some value $s_o > 0$. It is intuitively plausible that such a scheme could be adapted for mixtures of ellipsoidal Gaussians $\{N(\mu_i, \Sigma_i)\}$, under a similar separation condition, if σ_i were defined as the square root of the largest eigenvalue of Σ_i. This is because each $N(\mu_i, \Sigma_i)$ would then be contained within $N(\mu_i, \sigma_i^2 I_d)$, in the sense that the level sets of the former would lie within those of the latter. Such ideas have been fleshed out by Kannan et al. (2008) and Achlioptas and McSherry (2005).

This particular definition of σ_i is not ideal, however. It is equal to the (square root of the) *maximum* directional variance of the Gaussian, whereas a more natural and potentially much smaller quantity is the directional variance limited to the space spanned by the Gaussian means. There are several results of this type (Chaudhuri and Rao, 2008; Brubaker and Vempala, 2008; Chaudhuri et al., 2009), all of which make clever use of variants of PCA and have separation requirements depending only on k.

There are also some very general results that do not require any separation condition but have running times exponential in k and depending inversely on the separation (Belkin and Sinha, 2010; Kalai et al., 2012). The work of Kalai et al. (2012) is based on two fundamental observations:

1. A one-dimensional Gaussian mixture can be learned by doing a brute-force search through a suitable gridding of the parameter space and fitting the lowest $O(k)$ empirical moments. In particular, it can be shown that any mixture model with significantly different parameters will also differ significantly on at least one of these lower-order moments: a kind of *robust identifiability*.

2. A high-dimensional Gaussian mixture can be learned by taking multiple random projections to one dimension, solving the 1-d problems, and then suitably combining these solutions.

The independent work of Belkin and Sinha (2010) also involves the method of moments, and proceeds in greater generality. The authors define a parametrized family of distributions $\{p_\theta : \theta \in \Theta \subset \mathbb{R}^\ell\}$ to be a *polynomial family* if the individual distributions are identifiable from their moments, and if the moments of p_θ can be expressed as polynomial functions of θ. Many standard classes of distributions, including all exponential families and mixtures thereof, are polynomial families. It turns out that all such families have a robust identifiability property, which implies that a distribution can be accurately identified from low-order moments, whereupon a brute-force search through a quantized parameter space again suffices. The dependence of the running time on the dimension is reduced by finding a projection to a suitable low-dimensional subspace.

The procedures of Belkin and Sinha (2010) and Kalai et al. (2012) are to date the most general solutions to the problem of learning Gaussian mixture models. It is an open problem to obtain practical algorithms from their insights.

2.4 A nonparametric clustering model

So far we have talked about identifying a Gaussian mixture model given data sampled from it. This problem has been studied intensively, in part because it has been amenable to elegant analysis. However, the assumption of Gaussian clusters is very strong, particularly in high dimension, where it is common to have higher-order dependencies between variables.

We now move to a more general setting, where we merely assume that the data we see are sampled independently at random from some unknown density f on $\mathbb{R}^d$. Is there a notion of "natural" cluster structure in this case?

There are probably several reasonable such notions. One appealing option is the *cluster tree* associated with f (Hartigan, 1981). Specifically, a cluster is taken to be any connected component of $\{x : f(x) \geq \lambda\}$, for any $\lambda > 0$. Figure 4 shows three of the infinitely many clusters associated with a particular density on the line. As λ varies, the collection of all such clusters forms an (infinite) hierarchy, in the sense that for any $0 < \lambda' < \lambda$, each cluster at level λ is contained in a cluster at level λ'. We call this hierarchy the cluster tree $\mathbb{C}_f$. Formally, we define it as the function

$$\mathbb{C}_f : \lambda \mapsto \{R : R \text{ is a connected component of } \{x : f(x) \geq \lambda\}\}.$$

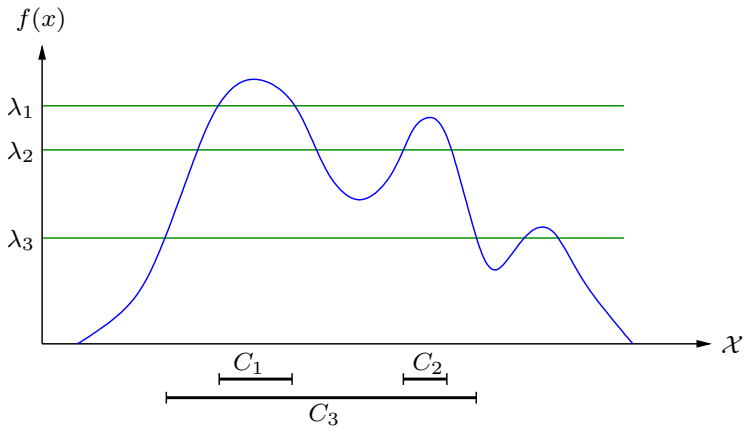


Figure 4: A probability density f on $\mathbb{R}$, and three of its clusters: C_1, C_2, and C_3.

Suppose we have samples $x_1, \ldots, x_n$ from f. We would like a procedure that builds a *hierarchical clustering* (or *dendrogram*) $\mathbb{C}_n$ from this data—a rooted tree whose leaves correspond to the individual x_i and each of whose subtrees corresponds to the cluster of its constituent leaves—with the property that as n grows, $\mathbb{C}_n$ converges to $\mathbb{C}_f$ in some suitable sense. We now formalize this requirement.

2.4.1 Notion of convergence

Let $\mathcal{X}$ be a subset of $\mathbb{R}^d$. We will exclusively consider Euclidean distance on $\mathcal{X}$, denoted $\|\cdot\|$. Let $B(x, r)$ be the closed ball of radius r around x.

Let f be a distribution on $\mathcal{X}$. Suppose a sample $X_n \subset \mathcal{X}$ of size n is used to construct a tree $\mathbb{C}_n$ that is an estimate of $\mathbb{C}_f$. Hartigan (1981) provided a sensible notion of consistency for this setting.

Definition 7 *For any sets $A, A' \subset \mathcal{X}$, let A_n (resp, A'_n) denote the smallest cluster of $\mathbb{C}_n$ containing $A \cap X_n$ (resp, $A' \cap X_n$). We say $\mathbb{C}_n$ is consistent if, whenever A and A' are different connected components of $\{x : f(x) \geq \lambda\}$ (for some $\lambda > 0$), $\Pr(A_n$ is disjoint from $A'_n) \to 1$ as $n \to \infty$.*

It is well known that if X_n is used to produce a uniformly consistent density estimate f_n, then the cluster tree $\mathbb{C}_{f_n}$ is consistent:

Lemma 8 *Suppose estimator f_n of density f (on space $\mathcal{X}$) satisfies*

$$\sup_{x \in \mathcal{X}} |f_n(x) - f(x)| \leq \epsilon_n.$$

Pick any two disjoint sets $A, A' \subset \mathcal{X}$ and define

$$\begin{aligned} \alpha &= \inf_{x \in A \cup A'} f(x) \\ \beta &= \sup \left\{ \inf_{x \in P} f(x) : P \text{ is a path from } A \text{ to } A' \right\} \end{aligned}$$

If $\alpha - \beta > 2\epsilon_n$ then A, A' lie entirely in disjoint connected components of $\mathbb{C}_{f_n}(\alpha - \epsilon_n)$.

PROOF: A and A' are each connected in $\mathbb{C}_f(\alpha)$ and thus also in $\mathbb{C}_{f_n}(\alpha - \epsilon_n)$. But there is no path from A to A' in $\mathbb{C}_{f_n}(\lambda)$ for $\lambda > \beta + \epsilon_n$. □

This does not constitute a viable solution because $\mathbb{C}_{f_n}$ is not easy to compute for typical density estimates f_n: imagine, for instance, how one might go about trying to find level sets of a mixture of Gaussians. Wong

and Lane (1983) have an efficient procedure that tries to approximate $\mathbb{C}_{f_n}$ when f_n is a k-nearest neighbor density estimate, but they have not shown that it preserves the consistency property of $\mathbb{C}_{f_n}$.

The problem, then, is to find an efficient and practical hierarchical clustering algorithm that is also consistent in the sense of Hartigan.

2.4.2 A hierarchical clustering procedure

There is a simple and elegant algorithm that is a plausible estimator of the cluster tree: *single linkage* or *Kruskal's algorithm* (Kruskal, 1956). This procedure for building a hierarchical clustering takes as input a data set $x_1, \ldots, x_n \in \mathbb{R}^d$, and proceeds as follows.

1. For each data point x_i, set $r_2(x_i) =$ distance from x_i to its nearest neighbor.
2. As r grows from 0 to ∞:
   (a) Construct a graph G_r with nodes $\{x_i : r_2(x_i) \leq r\}$.
   Include edge (x_i, x_j) if $\|x_i - x_j\| \leq r$.
   (b) Let $\widehat{\mathbb{C}}_n(r)$ be the connected components of G_r.

If an edge (x_i, x_j) is part of G_r, then it is also part of every subsequent $G_{r'}$ with $r' > r$. This implies, first, that the clusters produced by this algorithm form a hierarchy, and, second, that there are at most $\binom{n}{2}$ distinct graphs G_r. A careful implementation has a total running time of $O(n^2 \log n)$ (Tarjan, 1983).

Hartigan (1981) has shown that single linkage is consistent in one dimension ($d = 1$). But he also demonstrates, by a lovely reduction to continuum percolation, that this consistency fails in higher dimension $d \geq 2$. The problem is the requirement that $A \cap X_n \subset A_n$: by the time the clusters are large enough that one of them contains all of A, there is a reasonable chance that this cluster will be so big as to also contain part of A'.

With this insight, Hartigan defines a weaker notion of *fractional consistency*, under which A_n (resp, A'_n) need not contain *all* of $A \cap X_n$ (resp, $A' \cap X_n$), but merely a sizeable chunk of it; and ought to be very close (at distance $\to 0$ as $n \to \infty$) to the remainder. He then shows that single linkage has this weaker consistency property for any pair A, A' for which the quantity α/β (from Lemma 8) is sufficiently large. More recent work by Penrose (1995) closes the gap and shows fractional consistency whenever $\alpha/\beta > 1$.

A more robust version of single linkage has been proposed by Wishart (1969): when connecting points at distance r from each other, only consider points that have at least k neighbors within distance r (for some $k > 2$). Thus initially, when r is small, only the regions of highest density are available for linkage, while the rest of the data set is ignored. As r gets larger, more and more of the data points become candidates for linkage. This scheme has an intuitive appeal, but it is an open problem to determine whether it is consistent.

What we will see now is that a slight variant of single linkage, along the lines of Wishart's suggestion, is consistent and moreover admits finite-sample rates of convergence. The algorithm is shown in Figure 5. It has two free parameters: k and α. For practical reasons, it is of interest to keep these small. It is possible to show consistency for $\alpha > \sqrt{2}$ and $k \sim d \log n$; whether this also holds for $(\alpha = 1, k \sim d \log n)$ is open.

The results in this section are from Chaudhuri and Dasgupta (2010); further details can be found in that paper.

2.4.3 Rates of convergence

To get finite-sample rates of convergence, we need a notion of cluster salience: which clusters are easiest to pick out?

The first consideration is that a cluster is hard to identify if it contains a thin "bridge" that would make it look disconnected in a small sample. To control this, we consider a "buffer zone" of width σ around the clusters.

Definition 9 *For $Z \subset \mathbb{R}^d$ and $\sigma > 0$, write $Z_\sigma = Z + B(0, \sigma) = \{y \in \mathbb{R}^d : \inf_{z \in Z} \|y - z\| \leq \sigma\}$.*

1. For each x_i set $r_k(x_i) = \inf\{r : B(x_i, r) \text{ contains } k \text{ data points}\}$.
2. As r grows from 0 to ∞:
   (a) Construct a graph G_r with nodes $\{x_i : r_k(x_i) \leq r\}$. Include edge (x_i, x_j) if $\|x_i - x_j\| \leq \alpha r$.
   (b) Let $\widehat{\mathbb{C}}_n(r)$ be the connected components of G_r.

Figure 5: Algorithm for hierarchical clustering. The input is a sample $X_n = \{x_1, \ldots, x_n\}$. Parameters k and α need to be set. Single linkage is $(\alpha = 1, k = 2)$. Wishart suggested $\alpha = 1$ and larger k.

Z_σ is a full-dimensional set, even if Z itself is not.

Second, the ease of distinguishing two clusters A and A' depends inevitably upon the separation between them. To keep things simple, we'll use the same σ as a separation parameter.

Definition 10 *Let f be a density on $\mathcal{X} \subset \mathbb{R}^d$. We say that $A, A' \subset \mathcal{X}$ are* (σ, ϵ)-separated *if there exists $S \subset \mathcal{X}$ (*separator set*) such that:*

- *Any path in $\mathcal{X}$ from A to A' intersects S.*
- $\sup_{x \in S_\sigma} f(x) < (1 - \epsilon) \inf_{x \in A_\sigma \cup A'_\sigma} f(x)$.

Under this definition, A_σ and A'_σ must lie within $\mathcal{X}$, but S_σ might not.

With these notions in place, here is the convergence result for the algorithm of Figure 5.

Theorem 11 *There is an absolute constant C such that the following holds. Pick any $0 < \delta, \epsilon < 1$, and run the algorithm on a sample X_n of size n drawn from f, with settings*

$$\sqrt{2} \leq \alpha \leq 2 \quad \text{and} \quad k \geq C \cdot \frac{d \log n}{\epsilon^2} \cdot \log^2 \frac{1}{\delta}.$$

Then there is a decreasing function $r : [0, \infty) \to [0, \infty)$ such that with probability at least $1 - \delta$, the following holds uniformly for all pairs of connected subsets $A, A' \subset \mathcal{X}$: Suppose A, A' are (σ, ϵ)-separated (for ϵ and some $\sigma > 0$), and $\lambda = \inf_{x \in A_\sigma \cup A'_\sigma} f(x)$. If

$$n \;\geq\; \frac{k}{v_d (\sigma/2)^d \lambda} \left(1 + \frac{\epsilon}{2}\right)$$

where v_d is the volume of the unit ball in $\mathbb{R}^d$, then:

1. *Separation. $A \cap X_n$ is disconnected from $A' \cap X_n$ in $G_{r(\lambda)}$.*
2. *Connectedness. $A \cap X_n$ and $A' \cap X_n$ are each connected in $G_{r(\lambda)}$.*

These finite-sample results imply consistency (Definition 7): as $n \to \infty$, take $k_n = (d \log n)/\epsilon_n^2$ with any schedule of $(\epsilon_n : n = 1, 2, \ldots)$ such that $\epsilon_n \to 0$ and $k_n/n \to 0$. Under mild continuity conditions, any two connected components A, A' of $\{f \geq \lambda\}$ are (σ, ϵ)-separated for some $\sigma, \epsilon > 0$, and will thus get distinguished for sufficiently large n.

2.4.4 Analysis

In the mixtures of Gaussians setting, explicit knowledge of the distribution made it possible to give tight concentration bounds on interpoint distances. In the present setting, where the density f is arbitrary, we must rely upon more primitive forms of concentration. Since the class of balls in $\mathbb{R}^d$ has VC dimension just $d + 1$ (Dudley, 1979), it follows from standard large deviation bounds (Bousquet et al., 2004) that when the

sample size n is at least linear in d, every ball contains roughly its expected number of points. In particular, each ball $B(x, r_k(x))$ has empirical mass k/n by definition, and hence $f(B(x, r_k(x))) \approx k/n$.

We will also invoke uniform convergence over *half-balls*: each of these is the intersection of a ball with a halfspace passing through its center.

Lemma 12 *Assume $k \geq d \log n$, and fix some $\delta > 0$. Then there exists a constant C_δ such that with probability $> 1 - \delta$, we have that every ball (or half-ball) $B \subset \mathbb{R}^d$ satisfies the following conditions:*

$$\begin{aligned} f(B) \geq \frac{C_\delta d \log n}{n} &\implies f_n(B) > 0 \\ f(B) \geq \frac{k}{n} + \frac{C_\delta}{n}\sqrt{kd \log n} &\implies f_n(B) \geq \frac{k}{n} \\ f(B) \leq \frac{k}{n} - \frac{C_\delta}{n}\sqrt{kd \log n} &\implies f_n(B) < \frac{k}{n} \end{aligned}$$

Here $f_n(B) = |X_n \cap B|/n$ is the empirical mass of B, while $f(B)$ is its probability under f. We denote this uniform convergence over balls and half-balls as event E_o.

In order to show that two separate clusters A and A' get distinguished in the cluster tree, we need to exhibit a scale r at which G_r contains every point in A and A' but no path from A to A'. This r depends upon the density level λ of $A \cup A'$. To understand how it is chosen, observe that a ball of radius r in a region of density λ contains $\geq k$ points in expectation when $nv_d r^d \lambda \geq k$. To compensate for sampling error, we choose a slightly larger radius than this.

Definition 13 *For any $\lambda > 0$, define $r(\lambda)$ to be the value of r for which $nv_d r^d \lambda = k + C_\delta\sqrt{kd \log n}$.*

Lemma 14 (Separation) *Assume E_o. Suppose sets $A, A' \subset \mathcal{X}$ are (σ, ϵ)-separated by set S, and let $\lambda = \inf_{x \in A_\sigma \cup A'_\sigma} f(x)$. Suppose $r(\lambda) < 2\sigma/(\alpha + 2)$ and $k \geq 4C_\delta^2(d/\epsilon^2) \log n$. Then, for $r = r(\lambda)$:*

(a) G_r contains all points in $(A_{\sigma - r} \cup A'_{\sigma - r}) \cap X_n$.

(b) G_r contains no points in $S_{\sigma - r} \cap X_n$.

(c) $A \cap X_n$ is disconnected from $A' \cap X_n$ in G_r.

PROOF: For (a), any point $x \in A_{\sigma - r}$ (or $A'_{\sigma - r}$) has

$$f(B(x, r)) \;\geq\; v_d r^d \lambda \;\geq\; \frac{k}{n} + \frac{C_\delta}{n}\sqrt{kd \log n},$$

and thus, by Lemma 12, has at least k neighbors within radius r.

For (b), any point $x \in S_{\sigma - r}$ has

$$f(B(x, r)) \;<\; v_d r^d \lambda(1 - \epsilon) \;\leq\; \frac{k}{n} - \frac{C_\delta}{n}\sqrt{kd \log n},$$

and thus, by Lemma 12, has strictly fewer than k neighbors within distance r.

For (c), since points in $S_{\sigma - r}$ are absent from G_r, any path from A to A' in that graph must have an edge across $S_{\sigma - r}$. But any such edge has length at least $2(\sigma - r) > \alpha r$ and is thus not in G_r. □

To finish up, we need to show that points in A (and similarly A') are connected in $G_{r(\lambda)}$.

Lemma 15 (Connectedness) *Assume E_o. Let A be a connected set in $\mathcal{X}$ with $\lambda = \inf_{x \in A_\sigma} f(x)$. Suppose $\alpha \geq \sqrt{2}$, $k \geq (4C_\delta^2 d/\epsilon^2) \log n$, and $r(\lambda) \leq \sigma/2$. Then $A \cap X_n$ is connected in $G_{r(\lambda)}$.*

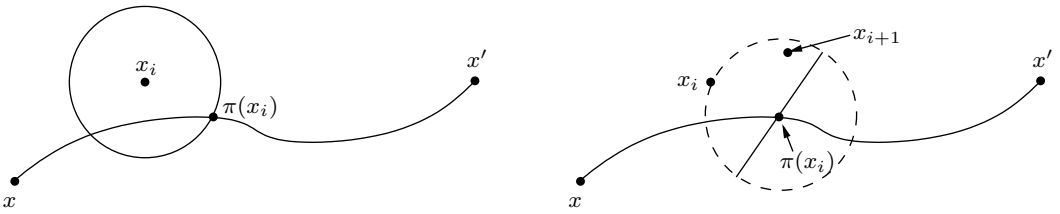


Figure 6: *Left:* P is a path from x to x', and $\pi(x_i)$ is the point furthest along the path that is within distance r of x_i. *Right:* The next point, $x_{i+1} \in X_n$, is chosen from the half-ball of $B(\pi(x_i), r)$ in the direction of $x_i - \pi(x_i)$.

PROOF: Take $r = r(\lambda)$. Pick any $x, x' \in A \cap X_n$; there is a path P in A from x to x'. We'll identify a sequence of data points $x_0 = x, x_1, x_2, \ldots$, ending in x', such that for every i, point x_i is active in G_r and $\|x_i - x_{i+1}\| \leq \alpha r$. This will confirm that x is connected to x' in G_r.

Think of P as a continuous function from $[0,1]$ into A. For any point $y \in \mathcal{X}$, define $N(y)$ to be the portion of $[0,1]$ whose image under P lies in $B(y,r)$: that is, $N(y) = \{0 \leq z \leq 1 : P(z) \in B(y,r)\}$. If y is within distance r of P, then $N(y)$ is nonempty. Define $\pi(y) = P(\sup N(y))$, the furthest point along the path within distance r of y (Figure 6, left).

The sequence $x_0, x_1, x_2, \ldots$ is defined iteratively; $x_0 = x$, and for $i = 0, 1, 2, \ldots$:

- If $\|x_i - x'\| \leq \alpha r$, set $x_{i+1} = x'$ and stop.
- By construction, x_i is within distance r of path P and hence $N(x_i)$ is nonempty.
- Let B be the open ball of radius r around $\pi(x_i)$. Under event E_o, the half-ball $B \cap \{z : (z - \pi(x_i)) \cdot (x_i - \pi(x_i)) > 0\}$ must contain a data point; this is x_{i+1} (Figure 6, right).

The process eventually stops because each $\pi(x_{i+1})$ is strictly further along path P than $\pi(x_i)$; formally, $\sup N(x_{i+1}) > \sup N(x_i)$. This is because $\|x_{i+1} - \pi(x_i)\| < r$, so by continuity of the function P, there are points further along the path (beyond $\pi(x_i)$) whose distance to x_{i+1} is still $< r$. Thus x_{i+1} is distinct from $x_0, x_1, \ldots, x_i$. Since there are finitely many data points, the process must terminate, so the sequence (x_i) does constitute a path from x to x'.

Each x_i lies in $A_r \subseteq A_{\sigma - r}$ and is thus active in G_r under event E_o (Lemma 14). Finally, the distance between successive points is:

$$\begin{aligned} \|x_i - x_{i+1}\|^2 &= \|x_i - \pi(x_i) + \pi(x_i) - x_{i+1}\|^2 \\ &= \|x_i - \pi(x_i)\|^2 + \|\pi(x_i) - x_{i+1}\|^2 - 2(x_i - \pi(x_i)) \cdot (x_{i+1} - \pi(x_i)) \\ &\leq 2r^2 \leq \alpha^2 r^2, \end{aligned}$$

where the second-last inequality is from the definition of the half-ball. □

Theorem 11 follows from Lemmas 14 and Lemmas 15.

2.4.5 Other work on density-based clustering

The results of this section are mostly from Chaudhuri and Dasgupta (2010); that paper also contains a lower bound on the convergence rate of any estimator. Another algorithm for estimating the cluster tree, based on *k-nearest neighbor graphs*, is given by Kpotufe and von Luxburg (2011).

There has been quite a lot of work of the *flat* version of this problem: recovering the connected components of $\{x : f(x) \geq \lambda\}$ for a user-specified λ (Rigollet and Vert, 2009; Maier et al., 2009; Singh et al., 2009; Rinaldo and Wasserman, 2010). These level set results do not directly yield a consistent estimator of the cluster tree because they impose various technical requirements on the specific level set being recovered, and it is not

clear how or when these requirements will be satisfied by all level sets of a distribution. A different approach is taken in a paper of Steinwart (2011), which does not require the user to specify a density level, but rather automatically determines the smallest λ at which $\mathbb{C}_f(\lambda)$ has two components. It also considers a broader class of distributions than other results in the literature.

Interestingly, the hierarchical setting resolves some of the technical hurdles of the flat case, because different scales appear at different levels of the tree, rather than being collapsed together.

2.4.6 Open problems on hierarchical clustering

There are a wide range of hierarchical clustering procedures used in practice, including agglomerative (bottom-up) schemes such as average linkage, and divisive (top-down) schemes based on repeated invocations of flat clustering subroutines. The convergence properties of these algorithms remain to be characterized.

We have discussed only one possible limiting object—the cluster tree—and only one possible notion of consistency, that due to Hartigan. This notion is sensible but in some ways too weak: it makes sure that distinct clusters get distinguished, but it does not guard against spurious branching (excessive fragmentation) in the estimated clustering. What is a stronger notion, and what procedures satisfy it? Some results in this direction have been obtained by Kpotufe and von Luxburg (2011) and Rinaldo et al. (2012).

Finally, what are other natural and desirable limits of hierarchical clustering procedures?

3 Exploiting low intrinsic dimensionality

Given data points of the form (X, Y), the *regression* problem is to infer the function $f(x) = \mathbb{E}[Y|X = x]$. *Nonparametric* models, such as kernel and tree-based regressors, allow an arbitrarily complex $f(\cdot)$ to be modeled while in many cases guaranteeing consistency (Gyorfi et al., 2002). The downside of this exceptional flexibility is a severe curse of dimension.

Suppose $X \in \mathbb{R}^d$, with marginal distribution μ. It is common practice to measure the error of an estimator f_n by its squared loss,

$$\mathbb{E}_{X\sim\mu}(f_n(X) - f(X))^2 \;=\; \int (f_n(x) - f(x))^2 \mu(dx).$$

Stone (1980, 1982) showed that this loss cannot in general be better than $n^{-2p/(2p+d)}$, where p is a parameter that captures the smoothness of f. If, for instance, f is only taken to be Lipschitz (that is, $|f(x) - f(x')| \leq L\|x - x'\|$ for some constant L) then $p = 1$. Regardless of the specific value of p, this means that to halve the error, the number of samples needs to grow by roughly a multiplicative factor of 2^d, which is prohibitive even for relatively small d.

Stone's lower bound would appear to rule out nonparametric approaches for the increasingly high-dimensional data sets that arise in modern applications. In image retrieval, or text classification, or genomic analysis, for instance, the number of features, or dimensions, of X can easily grow to tens of thousands, or more. However, in many of these cases, it is believed that the dimension is large only in the superficial sense of there being many coordinates, whereas the actual degrees of freedom are much smaller in number. This might occur, for example, because of strong dependencies between the features. It is therefore of interest to identify the *intrinsic dimension* of these data sets as the true measure of their complexity.

To take an example, a speech signal is typically represented by a high-dimensional time series: the signal is broken into overlapping windows, and a variety of filters is applied within each window. Even richer representations can be obtained by using more filters, or by concatenating vectors corresponding to consecutive windows. In this way, the dimension d can be made arbitrarily high. However, the physical system can alternatively be described by just a few ($d_o \ll d$) parameters specifying the configuration of the speaker's vocal apparatus. These are the true degrees of freedom of the data, and as they vary, the high-dimensional representation traces out a d_o-dimensional submanifold of $\mathbb{R}^d$. A recent trend in statistics and machine learning has been to design algorithms for data that lie on a manifold. Usually the goal is to recover the manifold, or else to obtain a mapping into a lower-dimensional space that preserves key quantities like interpoint distances.

A different type of low-dimensional structure arises in document classification. The most common way of representing a document is as a vector with one coordinate per word, which describes whether or not that word appears in the document—or the number of times the word appears, or some function thereof. The dimension d is therefore the size of the vocabulary, which is typically in the tens of thousands. However, any given document only contains a tiny fraction of these words, and thus most of its vector is zero: it is sparse. In a sense, the intrinsic dimension d_o of the data is the average number of non-zero entries, which is much smaller than d.

There are distinct types of low intrinsic dimension, as evidenced by the two examples above. It is therefore of interest to work with a broad definition that captures at least some of these, and to find nonparametric estimators that require resources depending on this quantity, rather than the apparent high dimension of the space in which the data lie.

3.1 Doubling dimension

There are many ways to formalize intrinsic dimension. One option that has worked well in both algorithmic and statistical work is the *doubling dimension*, which was introduced by Gupta et al. (2003) and is closely related to a dimensionality notion of Assouad (1983). It can be defined for any metric space, but in these notes we will only consider subsets of $\mathbb{R}^d$ under Euclidean distance.

Definition 16 *The doubling dimension of $\mathcal{X} \subset \mathbb{R}^d$ is the smallest d_o such that for any ball $B \subset \mathbb{R}^d$, the set $B \cap \mathcal{X}$ can be covered by 2^{d_o} balls of half the radius of B.*

Consider, for instance, a line S in a high-dimensional space $\mathbb{R}^d$. For any ball $B \subset \mathbb{R}^d$, the intersection of S and B is a line segment, and this segment can be covered by two balls whose radii are half that of B. Therefore the doubling dimension of S is 1. Something similar holds for any affine subspace of $\mathbb{R}^d$: as is well known, the unit ball in $\mathbb{R}^k$ has ϵ-covers of size $\leq (C/\epsilon)^k$ for some absolute constant $C < 3$, which implies the following.

Lemma 17 *There is a universal constant $c_o < 3$ such that for any $k < d$, a k-dimensional affine subspace of $\mathbb{R}^d$ has doubling dimension at most $c_o k$.*

Continuing with the theme of covering numbers, a set of diameter Δ and doubling dimension d_o can be covered by one ball of radius Δ, and thus 2^{d_o} balls of radius $\Delta/2$, and thus 2^{2d_o} balls of radius $\Delta/4$, and so on. Hence:

Lemma 18 *If $\mathcal{X}$ has diameter Δ and doubling dimension d_o, then for any $\epsilon > 0$, it has an ϵ-cover of size at most $(2\Delta/\epsilon)^{d_o}$.*

What makes the doubling dimension more than just a covering number for the overall space $\mathcal{X}$ is the following property, which is very helpful in the analysis of nonparametric estimators.

Lemma 19 *If $\mathcal{X}$ has doubling dimension d_o, then so does any subset of $\mathcal{X}$.*

Let's continue with some more examples. A set of n points can always be covered by n balls, and therefore has doubling dimension at most $\log_2 n$. This is easily generalized:

Lemma 20 *Suppose sets $S_1, \ldots, S_n$ each have doubling dimension $\leq d_o$. Then $S_1 \cup \cdots \cup S_n$ has doubling dimension at most $d_o + \log_2 n$.*

We can now get a bound on the doubling dimension of any sparse set.

Lemma 21 *Suppose that $S \subset \mathbb{R}^d$ is k-sparse: that is, each point in S has at most k nonzero coordinates. Then S has doubling dimension at most $c_o k + k \log d$.*

PROOF: S is contained within the union of $\binom{d}{k} \leq d^k$ subspaces of dimension k: pick which k coordinates, out of d, will be nonzero, and consider the subspace in which the remaining coordinates are forced to zero. By Lemma 17, each of these subspaces has doubling dimension at most $c_o k$. Lemma 20 then bounds the increase in dimension from taking the union of the subspaces. □

Thus the doubling dimension captures sparse data, a subject of significant contemporary interest. What about manifold data? Here the situation is slightly more subtle. Although it is intuitively sensible in many situations to suppose that data lie on (or close to) a low-dimensional manifold, this is not of much help, algorithmically or statistically, unless the manifold has bounded curvature: a space-filling 1-dimensional curve, for instance, is just as bad as a full-dimensional data set. The work of Niyogi et al. (2006) captures curvature by a single value called the *condition number* of the manifold; it is closely related to the notion of *local feature size* in the computational geometry literature (Amenta and Bern, 1999). When this is bounded, neighborhoods of the manifold are sufficiently flat that they can be shown to have low doubling dimension (Dasgupta and Freund, 2008).

Lemma 22 *If a k-dimensional Riemannian submanifold of $\mathbb{R}^d$ has bounded condition number $\tau < \infty$, then its neighborhoods of radius $< 1/\tau$ have doubling dimension $O(k)$.*

In fact, the containment is strict: there is a substantial gap between manifolds of bounded curvature and sets of low doubling dimension, on account of the smoothness properties of the former. This divide has many algorithmic consequences. For instance, a variant of the Johnson-Lindenstrauss lemma (Johnson and Lindenstrauss, 1984) states that when a d_o-dimensional manifold (of bounded curvature) is projected into a random subspace of dimension $O(d_o/\epsilon^2)$, then all interpoint distances are preserved within $1 \pm \epsilon$ (Baraniuk and Wakin, 2009; Clarkson, 2008). For sets of doubling dimension d_o, however, no such guarantee can be given: an arbitrarily high-dimensional range space might be needed (Indyk and Naor, 2007).

3.2 Finding embeddings of spaces of low intrinsic dimension

If a data distribution is supported on some $\mathcal{X} \subset \mathbb{R}^d$ of intrinsic dimension d_o, one option is to use samples from the distribution to construct a mapping $\Phi : \mathbb{R}^d \to \mathbb{R}^k$ such that:

- k is much smaller than d, ideally $k = O(d_o)$,
- Φ is one-to-one on $\mathcal{X}$, and
- Φ roughly preserves the neighborhood structure of $\mathcal{X}$, in some suitable sense.

Estimation tasks like regression can then be undertaken in the smaller space.

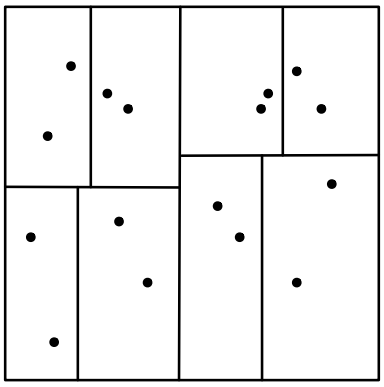
A variety of elegant algorithms have been developed for this embedding task in the specific case where $\mathcal{X}$ is a d_o-dimensional manifold; see, for instance, Tenenbaum et al. (2000), Roweis and Saul (2000), and Belkin and Niyogi (2003). Some of these methods have partial consistency results, but it is still not entirely clear what can be expected of them given realistic amounts of data sampled from realistic distributions. All in all, the embedding problem seems difficult.

Is there a simpler representation of $\mathcal{X}$ that is easy to construct and yet provably adapts to intrinsic dimension? The obvious candidate is a tree-based spatial partition.

3.3 Spatial partitioning for nonparametric estimation

A *spatial partition tree* is a recursive partitioning of an instance space $\mathcal{X} \subset \mathbb{R}^d$. Initially there is a single cell containing the entire space; then it is split into two cells; then each of these is split again, and so on. A popular such data structure is the k-d tree (Figure 7), which splits a cell by choosing a coordinate direction and splitting at the median along that direction. The resulting partitioning consists of hyperrectangular regions.

Once such a hierarchical partition has been constructed, it can be used for classification, regression, and other statistical tasks. In tree-based regression or classification (Breiman et al., 1984), it is common to fit



```
function MakeTree(S)
  If |S| < MinSize: return (Leaf)
  Rule = ChooseRule(S)
  LeftTree = MakeTree{x ∈ S : Rule(x) = true}
  RightTree = MakeTree{x ∈ S : Rule(x) = false}
  return (Rule, LeftTree, RightTree)

function ChooseRule(S)
  Choose a coordinate direction i
  Rule(x) = (x_i ≤ median({z_i : z ∈ S}))
  return (Rule)
```

Figure 7: The k-d tree: an example, and pseudocode for construction.

a simple model to the data in each leaf of the tree: for instance, a constant, or a linear function. To get a prediction at a new point x, that point is moved down the tree to its leaf node, taking time proportional to the height of the tree, and then the model in that leaf is applied. The overall simplicity and efficiency of this scheme is one of its great attractions.

A detailed overview of the generalization theory for tree-based regression and classification can be found in the texts of Devroye et al. (1996) and Gyorfi et al. (2002). Roughly speaking, consistency is obtained if, as the number of samples goes to infinity,

1. the diameter of the leaf cells shrinks to zero, and
2. the number of samples falling in each leaf cell increases to infinity.

If the regression function $\mathbb{E}[Y|X = x]$ is smooth in some suitable sense—say, Lipschitz—then rates of convergence can also be given, based on the relative speed of these two effects.

Let's take the k-d tree, for example. If we think of the root as level 0, then level ℓ of the tree has 2^ℓ nodes (or cells), each containing a $1/2^\ell$ fraction of the data. How fast does the diameter of these cells shrink down the tree?

It shrinks rather slowly, in general. Consider an extremely sparse data set $S \subset \mathbb{R}^d$, made up of the coordinate axes between -1 and 1:

$$S = \bigcup_{i=1}^{d} \{te_i : -1 \leq t \leq 1\}.$$

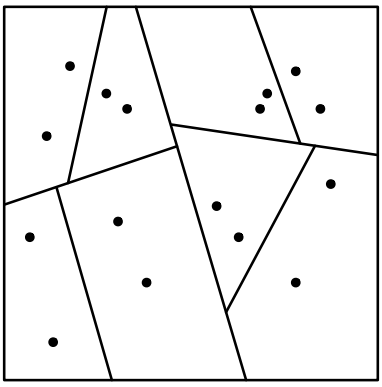
Here $e_1, \ldots, e_d$ is the canonical basis of $\mathbb{R}^d$. The diameter of S is two, and it is not hard to see that the k-d tree would require at least d levels in order to halve this diameter. This yields the usual curse-of-dimension $n^{-1/\Omega(d)}$ convergence rate. Notice, however, that the doubling dimension of S is just $d_o = \log 2d$; the k-d tree is not adaptive to this kind of intrinsic dimension.

On the other hand, we will now see that a slight variant of the k-d tree *is* adaptive: when the doubling dimension is d_o, the diameter of cells is halved every $O(d_o \log d_o)$ levels of the tree.

3.4 Random projection trees

Like k-d trees, random projection (RP) trees are built by recursive binary splits. They differ only in the nature of the split. Instead of splitting along a coordinate direction, an RP tree chooses a direction uniformly at random from the unit sphere S^{d-1}; and instead of splitting exactly at the median along this direction, it adds a small random perturbation to the median (Figure 8).

Suppose an RP tree is built from a data set $S \subset \mathbb{R}^d$, not necessarily finite. If the tree has k levels, then it partitions the space into 2^k cells. We define the *radius* of a cell $C \subset \mathbb{R}^d$ to be the smallest $r > 0$ such that



function ChooseRule(S)
Choose a random unit direction $v \in \mathbb{R}^d$
Pick any $x \in S$; let $y \in S$ be the farthest point from it
Choose δ uniformly at random in $[-1, 1] \cdot 6\|x - y\|/\sqrt{d}$
Rule(x) = $x \cdot v \leq (\text{median}(\{z \cdot v : z \in S\}) + \delta)$
return (Rule)

Figure 8: The RP tree: example and pseudocode.

$S \cap C \subset B(x, r)$ for some $x \in C$. The following theorem from Dasgupta and Freund (2008) gives an upper bound on the rate at which the radius of cells in an RP tree decreases as one moves down the tree.

Theorem 23 *There is a constant c_1 with the following property. Suppose an RP tree is built using data set $S \subset \mathbb{R}^d$. Pick any cell C in the RP tree; suppose that $S \cap C$ has doubling dimension $\leq d_o$. Then with probability at least $1/2$ (over the randomization in constructing the subtree rooted at C), for every descendant C' which is more than $c_1 d_o \log d_o$ levels below C, we have* $\text{radius}(C') \leq \text{radius}(C)/2$.

We now give a brief proof sketch; for further details see the original paper.

3.4.1 Overview of proof

Suppose an RP tree is built using data set $S \subset \mathbb{R}^d$ of doubling dimension d_o, and that C is some cell of the tree. If $S \cap C$ lies in a ball of radius Δ, then we need to show that after $O(d_o \log d_o)$ further levels of splitting, each resulting descendant of C is contained in a ball of radius $\leq \Delta/2$.

1. Consider a cover of $S \cap C$ by balls $B_1, \ldots, B_N$ of radius $\Delta/\sqrt{d_o}$. By Lemma 18, $N = O(d_o)^{d_o/2}$ suffices.
2. Pick any two balls B_i and B_j whose centers are more than $(\Delta/2) - (\Delta/\sqrt{d_o})$ apart. It can be shown that a single random split has a constant probability of cleanly separating them, in the sense that B_i and B_j will lie entirely on opposite sides of the split.
3. There are at most N^2 such pairs i, j, so with probability at least $1/2$, after $O(d_o \log d_o)$ splits every pair will have been separated. In this event, $O(d_o \log d_o)$ levels below C in the tree, each cell will only contain points from balls B_i which are within distance $(\Delta/2) - (\Delta/\sqrt{d_o})$ of each other. Hence the radius of these cells will be $\leq \Delta/2$.

We now delve into some of the details of step (2). For the two balls B_i and B_j, let $\Pi(B_i)$ and $\Pi(B_j)$ denote their projections onto a randomly chosen direction. It can be shown that with constant probability the following events occur:

(a) $\Pi(B_i)$ and $\Pi(B_j)$ are separated by an interval of length $\Omega(1/\sqrt{d})$.

(b) The median of the projected data lies within $O(1/\sqrt{d})$ of this separating interval.

(c) The split point, chosen randomly from an interval around the median, separates $\Pi(B_i)$ from $\Pi(B_j)$.

To understand these effects, we need to look at some basic characteristics of the projection.

3.4.2 Gross statistics of projected data

We choose random projections from $\mathbb{R}^d$ to $\mathbb{R}$ as follows:

- Pick U from the multivariate Gaussian $N(0, (1/d)I_d)$.
- Define projection $\Pi(x) = U \cdot x$.

Notice that $\Pi(x)$ also has a Gaussian distribution, $N(0, \|x\|^2/d)$, on account of which we can immediately assert the following.

Lemma 24 *Fix any $x \in \mathbb{R}^d$. Pick a random projection Π as above. Then for any $\alpha, \beta > 0$:*

(a) $\Pr\left[|\Pi(x)| \leq \alpha \cdot \frac{\|x\|}{\sqrt{d}}\right] \leq \sqrt{\frac{2}{\pi}}\,\alpha$; *and*

(b) $\Pr\left[|\Pi(x)| \geq \beta \cdot \frac{\|x\|}{\sqrt{d}}\right] \leq \frac{2}{\beta}e^{-\beta^2/2}$.

In particular, the projection approximately preserves the lengths of *individual* vectors, modulo a scaling factor of $\sqrt{d}$. What about the behavior of the projected *ensemble* of points: where do most of them lie, what is their diameter, and what is their median?

Suppose $S \subset \mathbb{R}^d$ has doubling dimension d_o. Let $\Pi(S) = U \cdot S$ be its random projection into $\mathbb{R}$. How does $\mathrm{diam}(\Pi(S))$ compare to $\mathrm{diam}(S)$? (Here $\mathrm{diam}(S) = \sup_{x,y\in S}\|x-y\|$.) Clearly $\mathrm{diam}(\Pi(S)) \leq \|U\| \cdot \mathrm{diam}(S)$, but we would in fact expect it to be much smaller. In fact, $\mathrm{diam}(\Pi(S)) \leq \mathrm{diam}(S) \cdot O(\sqrt{d_o/d})$:

Lemma 25 *Suppose set $S \subset \mathbb{R}^d$ is contained in a ball $B(x_0, \Delta)$ and has doubling dimension d_o. Let $\Pi(S)$ denote the random projection of S into $\mathbb{R}$. Then for any $0 < \delta < 1$, with probability $> 1 - \delta$ over the choice of projection, $\Pi(S)$ lies in an interval of radius $4 \cdot \frac{\Delta}{\sqrt{d}} \cdot \sqrt{2\left(d_o + \ln\frac{2}{\delta}\right)}$ centered at $\Pi(x_0)$.*

Proof: Without loss of generality, take $\Delta = 1$. Here we'll just sketch the proof of a weaker version of this lemma, showing that $\Pi(S)$ has diameter $O(\sqrt{(d_o \log d)/d})$. The extra $\log d$ factor can be shaved off by a more careful multiscale argument.

Cover S by balls $B_1, \ldots, B_N$ of radius $\sqrt{1/d}$; by Lemma 18, at most $N = (4d)^{d_o/2}$ balls are needed. For any such ball B_i, its center lies within distance 1 of x_0, and thus, by Lemma 24(b), the probability that this center gets projected to a point further than $\sqrt{(cd_o \ln d)/d}$ from $\Pi(x_o)$ is at most $d^{-cd_o/2}$, for any $c > 1$. Taking a union bound over the balls, we have that with probability at least $1 - Nd^{-cd_o/2}$ all the $\Pi(B_i)$ lie within distance $\sqrt{(cd_o \ln d)/d} + \sqrt{1/d}$ of $\Pi(x_0)$, and thus $\Pi(S)$ is contained within this interval. □

Thus, S projects to an interval in $\mathbb{R}$ of radius at most $O(\Delta \cdot \sqrt{d_o/d})$. In fact, it follows immediately from Lemma 24 that most of the projected points will be even closer together, in a *central interval* of size $O(\Delta/\sqrt{d})$.

Lemma 26 *Suppose $S \subset \mathbb{R}^d$ lies within ball $B(x_0, \Delta)$. Pick any $0 < \delta, \epsilon \leq 1$ such that $\delta\epsilon \leq 1/e^2$. Let μ be any probability measure on S. Then with probability $> 1 - \delta$ over the choice of random projection onto $\mathbb{R}$, all but an ϵ fraction of $\Pi(S)$ (in μ-measure) lies within distance $\sqrt{2\ln\frac{1}{\delta\epsilon}} \cdot \frac{\Delta}{\sqrt{d}}$ of $\Pi(x_0)$.*

As a consequence, the median of the projected points will also lie in this central interval; take μ to be the uniform distribution over S and use $\epsilon = 1/2$.

Corollary 27 *Under the hypotheses of Lemma 26, for any $0 < \delta < 2/e^2$, with probability at least $1 - \delta$ over the choice of projection: $|\mathrm{median}(\Pi(S)) - \Pi(x_0)| \leq \frac{\Delta}{\sqrt{d}} \cdot \sqrt{2\ln\frac{2}{\delta}}$.*

3.4.3 The probability of a clean split

Recall that the split rule looks at a random projection of the data and then splits it *approximately* at the median. The perturbation added to the median depends on the diameter of the space.

Lemma 28 *Say $S \subset B(x_0, \Delta)$ has doubling dimension $d_o \geq 1$. Pick balls $B = B(z, r)$ and $B' = B(z', r)$ such that*

- *their centers z and z' lie in $B(x_0, \Delta)$,*
- *the distance between these centers is $\|z - z'\| \geq \frac{1}{2}\Delta - r$,*
- *and the radius r is at most $\Delta/(512\sqrt{d_o})$.*

Now pick a random projection Π, and then pick a split point at random in the range $\text{median}(\Pi(S)) \pm (6\Delta/\sqrt{d})$. With probability at least $1/192$ over the choice of U and the split point, $S \cap B$ and $S \cap B'$ will be contained in separate halves of the split.

PROOF: (Sketch.) Let $\Pi(B)$ and $\Pi(B)'$ be the projections of $S \cap B$ and $S \cap B'$. With probability at least $1/2$, the random projection Π will satisfy the following properties:

1. $\Pi(B)$ and $\Pi(B')$ are contained within intervals of radius at most $\Delta/(16\sqrt{d})$ around $\Pi(z)$ and $\Pi(z')$, respectively (Lemma 25).
2. $|\Pi(z) - \Pi(z')| \geq \Delta/(4\sqrt{d})$ (Lemma 24(a)).
3. $\Pi(z)$ and $\Pi(z')$ both lie within distance $3\Delta/\sqrt{d}$ of $\Pi(x_0)$ (Lemma 24(b)).
4. The median of $\Pi(S)$ lies within distance $3\Delta/\sqrt{d}$ of $\Pi(x_0)$ (Corollary 27).

In this event, there is an interval of length at least $\Delta/(8\sqrt{d})$ between $\Pi(B)$ and $\Pi(B')$, and if the split point falls in it, the two balls will be cleanly separated.

Moreover, by (3) and (4), we know that the entirety of the interval lies within distance $3\Delta/\sqrt{d}$ of $\Pi(x_0)$ and thus within distance $6\Delta/\sqrt{d}$ of $\text{median}(\Pi(S))$. Thus, there is a constant probability that the split point will fall in this interval. □

Theorem 23 now follows, as outlined in the initial overview.

3.5 A nonparametric regressor that is adaptive to doubling dimension

Suppose data points (X, Y) are drawn i.i.d. from an unknown distribution on $\mathcal{X} \times \mathcal{Y} \subset \mathbb{R}^d \times \mathbb{R}^{d'}$. Let μ denote the marginal distribution of X, and let $f(x) = \mathbb{E}[Y|X = x]$ be the regression function. We wish to estimate this latter function.

It is customary to assess an estimate $g : \mathcal{X} \to \mathcal{Y}$ of f by its *integrated excess risk*

$$\|f - g\|^2 = \int \|f(x) - g(x)\|^2 \mu(dx) = \mathbb{E}_X \|f(X) - g(X)\|^2.$$

The rate of convergence of any such estimator will depend on how smooth f is, and there are a variety of ways in which this can be quantified. Here we simply assume that f is λ-Lipschitz for some (unknown) λ:

$$\forall x, x' \in \mathcal{X},\ \|f(x) - f(x')\| \leq \lambda \|x - x'\|.$$

Without further assumptions, the best possible rate of convergence is $n^{-1/\Omega(d)}$ (Stone, 1980). We are interested in situations where the domain $\mathcal{X}$ has low doubling dimension $d_o < d$, and we will see that it is then possible to obtain rates in which d_o takes the place of d.

Suppose the training set consists of n points $S = \{X_1, \ldots, X_n\}$ along with corresponding response values $Y_1, \ldots, Y_n$. Let μ_n denote the empirical distribution over $\mathcal{X}$ that assigns mass $1/n$ to each X_i. We will consider a specific estimator that first builds a spatial partition tree using S alone, and then brings in the Y_i to fit models to each leaf of the tree.

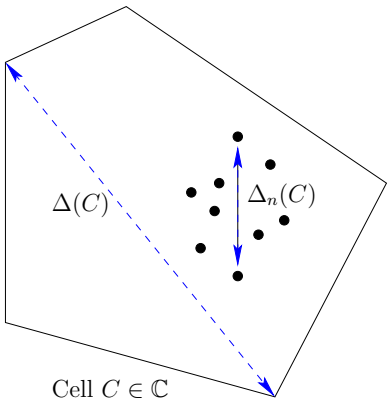


Figure 9: Physical diameter $\Delta(C)$ versus data diameter $\Delta_n(C)$.

3.5.1 Notions of diameter

A typical tree-based regressor works in two phases.

1. The data is used to construct a partition $\mathbb{C}$ of $\mathcal{X}$.
2. A regressor is learned as a piecewise continuous function over the cells of $\mathbb{C}$.

For instance, a piecewise constant regressor over $\mathbb{C}$ is defined as follows: for any $x \in \mathcal{X}$, let $\mathbb{C}(x)$ be the cell of $\mathbb{C}$ to which x belongs, and define

$$f_{n,\mathbb{C}}(x) = \frac{\sum_{i=1}^{n} Y_i \cdot \mathbf{1}\left(X_i \in \mathbb{C}(x)\right)}{n \cdot \mu_n(\mathbb{C}(x))} \tag{2}$$

if $\mu_n(\mathbb{C}(x)) > 0$ (that is, if the cell $\mathbb{C}(x)$ contains at least one training point). If $\mathbb{C}(x) \cap S$ is empty, then a default setting $f_{n,\mathbb{C}}(x) = y_o$ is used instead, for some $y_o \in \mathcal{Y}$. We will often refer to the final regressor as f_n when the partition $\mathbb{C}$ used for the estimate is clear from context.

In analyzing the consistency of such estimators (Gyorfi et al., 2002), it is standard to decompose the risk into two parts, the *bias* (how much does f vary within a single cell?) and the *variance* (what is the error in estimating the mean value of f within a cell?).

The bias can be controlled by making sure cells are small. Traditionally, this analysis has been based on the *physical diameters* of cells $C \in \mathbb{C}$,

$$\Delta(C) = \max_{x,x' \in C} \|x - x'\|;$$

see, for instance, Devroye et al. (1996). These are easy to control when the cells are regular, for instance if they are hyperrectangles. But we wish to handle richer partitioning schemes in which the cells have arbitrary shapes. Therefore, we will instead relate bias to the *data diameters* of the cells,

$$\Delta_n(C) = \max_{x,x' \in C \cap S} \|x - x'\|$$

(or 0 if $C \cap S$ is empty); recall that S is the set of data points. See Figure 9.

In fact, an even weaker notion can be used. It is not necessary for all cells of a partition to have small data diameter, but merely for these diameters to be small in an average sense. For a collection $\mathbb{C}$ of disjoint subsets of $\mathcal{X}$, we use the following notion of average data diameter:

$$\Delta_n(\mathbb{C}) \quad = \quad \sqrt{\frac{\sum_{C \in \mathbb{C}} \mu_n(C) \Delta_n^2(C)}{\sum_{C \in \mathbb{C}} \mu_n(C)}}.$$

$\mathbb{C}_0 = \mathbb{R}^d$
Define $\alpha(n) = (\log^2 n)\log\log(n/\delta) + \log(1/\delta)$
For $i = 1, 2, \ldots$:
  For each cell $C \in \mathbb{C}_{i-1}$:
    Set the subtree rooted at C to `coreRPtree`$(C \cap S)$
  Let $\mathbb{C}_i$ be the partition of $\mathbb{R}^d$ defined by the leaves of the current tree
  If $\Delta_n^2(\mathbb{C}_i) \leq \Delta_n^2(\mathbb{C}_0) \cdot (\alpha(n)/n) \cdot 2^{\text{depth}(\mathbb{C}_i)}$:
    Let $\mathbb{C}^*$ be either $\mathbb{C}_{i-1}$ or $\mathbb{C}_i$, whichever has smaller $\left(\frac{\alpha(n)}{n} \cdot |\mathbb{C}| + \Delta_n^2(\mathbb{C})\right)$
    Return $f_{n,\mathbb{C}^*}$

Figure 10: Constructing a regression tree given a sample of n points and a confidence parameter δ. Subroutine `coreRPtree` is described in the text.

These changes broaden the class of estimators for which finite-sample risk bounds can be given, but they also necessitate significant changes to the usual style of analysis. For concreteness, in this section we look specifically at regressors based on random projection trees. The results we present are from Kpotufe and Dasgupta (2012); further details can be found in that paper.

3.5.2 Regression based on RP trees

Figure 10 shows a procedure for constructing a type of random projection tree, with a stopping rule that prevents it from getting too large. It starts with a single node $\mathbb{C}_0$ for all of $\mathbb{R}^d$, and then grows the tree in measured steps. At each stage, the current set of leaves constitute a partition $\mathbb{C}_i$ of $\mathbb{R}^d$, whose cells have average diameter $\Delta_n(\mathbb{C}_i) \leq 2^{-i}\Delta_n(\mathbb{R}^d)$. Then `coreRPtree` is called on each leaf to yield an even finer partition $\mathbb{C}_{i+1}$. This subroutine takes as input a cell $C \subset \mathbb{R}^d$ (more precisely, the data points that fall in that cell), and behaves as follows:

- It returns a subtree whose root corresponds to C and whose bottom level has average data diameter Δ_n at most half that of C.
- If C has zero diameter (for instance, if it contains one point), then the procedure leaves it untouched.
- If the subtree has height ℓ, then its leaves each contain $\leq \lceil |C \cap S|/2^{\ell/2} \rceil$ points.

The main procedure stops growing the tree when each cell of the current partition is sufficiently small that the bias is controlled, but also has sufficiently many data points in it that the variance is controlled: that is, when the average data diameter of cells is small enough relative to the tree size. By the properties of `coreRPtree`, the final tree has height at most $2\log 2n$ and the number of partitions $\mathbb{C}_i$ generated is at most $\log 2n$.

Once the tree is built, and a final partition $\mathbb{C}^*$ is obtained, the response values Y_i are used to define a piecewise constant regressor $f_{n,\mathbb{C}^*}$ as in (2). The risk of this regressor can be expressed in terms of the rate at which cell diameters decrease from the root down. We adopt the following definition.

Definition 29 *We say that* `coreRPtree` *attains a diameter decrease rate of k on sample S, if every call to it in the main procedure returns a subtree of depth at most k.*

Using RP tree splits, a diameter decrease rate of $O(d_o \log d_o)$ can be achieved, where d_o is the doubling dimension of $\mathcal{X}$. Building upon this result, we have the following main theorem.

Theorem 30 *Assume that $\mathcal{X}$ has doubling dimension d_o. There exist constants A, A' independent of d, d_o, and μ, such that the following hold. Pick any $0 < \delta < 1$ and define $\alpha(n) = (\log^2 n)\log\log(n/\delta) + \log(1/\delta)$. With probability at least $1 - \delta$, we have that*

(a) `coreRPtree` *attains a diameter decrease rate of $k \leq A' d_o \log d_o$, and*

(b) the excess risk of the regressor is

$$\|f_n - f\|^2 \leq A \cdot \left(\Delta_{\mathcal{Y}}^2 + \lambda^2\right)\left(\Delta_{\mathcal{X}}^2 + 1\right) \cdot \left(\frac{\alpha(n)}{n}\right)^{2/(2+k)}.$$

where $\Delta_{\mathcal{X}}$ and $\Delta_{\mathcal{Y}}$ are the diameters of spaces $\mathcal{X}$ and $\mathcal{Y}$, respectively.

The overall failure probability δ is divided between three contingencies: the tree might not achieve the desired diameter decrease rate; the empirical masses of cells might not accurately represent their true masses; and the y-values within cells might have non-representative averages.

We now give a brief overview of the proof technique.

3.5.3 An alternate partition for analyzing risk

We start the analysis with a standard decomposition of the excess risk into bias and variance terms. Let $\mathbb{C}$ be any partition of $\mathcal{X}$, and define the regressor $f_{n,\mathbb{C}}$ as in (2). Recall that $\mathbb{C}(x)$ is the cell of $\mathbb{C}$ containing x.

A useful intermediary between $f_{n,\mathbb{C}}$ and the target f is the following function on $\mathcal{X}$:

$$\widetilde{f}_{n,\mathbb{C}}(x) \quad = \quad \frac{\sum_{i=1}^n f(X_i)\mathbf{1}\left(X_i \in \mathbb{C}(x)\right)}{n\mu_n(\mathbb{C}(x))}$$

if $\mu_n(\mathbb{C}(x)) \neq 0$; otherwise $\widetilde{f}_{n,\mathbb{C}}(x) = y_o \in \mathcal{Y}$. Notice that both $f_{n,\mathbb{C}}$ and $\widetilde{f}_{n,\mathbb{C}}$ are constant within any cell $C \in \mathbb{C}$; we will occasionally write these quantities as $f_{n,\mathbb{C}}(C)$ and $\widetilde{f}_{n,\mathbb{C}}(C)$, respectively.

The pointwise excess risk at x can be bounded as

$$\begin{aligned}
\|f_{n,\mathbb{C}}(x) - f(x)\|^2 &\leq \left(\left\|f_{n,\mathbb{C}}(x) - \widetilde{f}_{n,\mathbb{C}}(x)\right\| + \left\|\widetilde{f}_{n,\mathbb{C}}(x) - f(x)\right\|\right)^2 \\
&\leq \underbrace{2\left\|f_{n,\mathbb{C}}(\mathbb{C}(x)) - \widetilde{f}_{n,\mathbb{C}}(\mathbb{C}(x))\right\|^2}_{\text{variance}} + \underbrace{2\left\|\widetilde{f}_{n,\mathbb{C}}(x) - f(x)\right\|^2}_{\text{bias}^2}. \qquad (3)
\end{aligned}$$

The next two lemmas are standard. The first bounds the variance within a cell, using McDiarmid's concentration inequality, while the second bounds the bias in terms of cell diameter, using the Lipschitz condition on f.

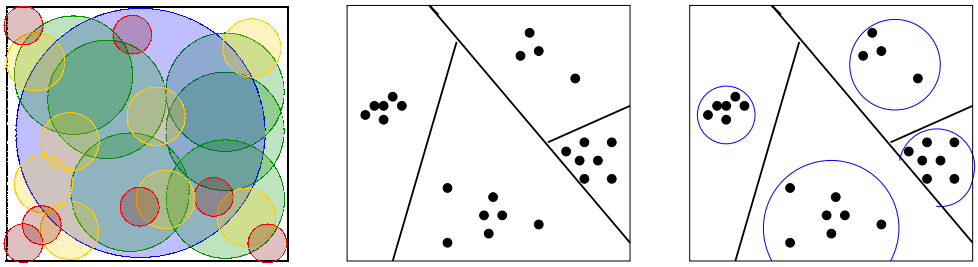
Lemma 31 (Variance) *Fix any partition $\mathbb{C}$ and any set of n points $S = \{X_1, \ldots, X_n\} \subset \mathcal{X}$. Suppose the Y_i are now drawn according to their conditional distribution given X_i. Pick any $0 < \delta < 1$. Then with probability at least $1 - \delta$, for every cell $C \in \mathbb{C}$ with $\mu_n(C) > 0$:*

$$\left\|f_{n,\mathbb{C}}(C) - \widetilde{f}_{n,\mathbb{C}}(C)\right\|^2 \leq \Delta_{\mathcal{Y}}^2 \cdot \frac{2 + \ln(|\mathbb{C}|/\delta)}{n\mu_n(C)}.$$

Lemma 32 (Bias) *For any $x \in \mathcal{X}$ with $\mu_n(\mathbb{C}(x)) > 0$, we have $\|\widetilde{f}_{n,\mathbb{C}}(x) - f(x)\| \leq \lambda \cdot \Delta\left(\mathbb{C}(x)\right)$.*

The bound on bias involves the physical diameters $\Delta(C)$ of cells, which might not decrease gracefully down the tree and in any case seem complicated to control. To deal with this, we introduce an alternate partition $\mathbb{C}'$ that helps in analyzing the regressor based on $\mathbb{C}$. It is designed so that $f_{n,\mathbb{C}'}$ is equivalent to $f_{n,\mathbb{C}}$ on most of $\mathcal{X}$, and in addition has the following properties:

1. Each cell of $\mathbb{C}$ is the union of two cells of $\mathbb{C}'$.
2. Every cell in $\mathbb{C}'$ is either void of data points (and thus likely has low probability under μ and can be disregarded) or else has a physical diameter that is roughly the same as its data diameter.



(a) Cover $\mathcal{B}$ (b) Partition $\mathbb{C}$ (c) Partition $\mathbb{C}'$

Figure 11: We start with a cover $\mathcal{B}$ of $\mathcal{X}$ with balls of different size; then, we see the data and obtain a partition $\mathbb{C}$; and finally we substitute $\mathbb{C}$ with an alternate partition $\mathbb{C}'$, by intersecting the cells of $\mathbb{C}$ with balls of $\mathcal{B}$.

The second property makes it possible to move between the physical diameters of cells and their data diameters.

$\mathbb{C}'$ is obtained by intersecting the cells of $\mathbb{C}$ with balls or complements-of-balls from a fixed, pre-defined collection $\mathcal{B}$ (Figure 11). Specifically, let $\mathcal{B}_i$ be a cover of $\mathcal{X}$ by balls of radius $\Delta_{\mathcal{X}}/2^i$. Take a variety of scales: $i = 0, 1, 2, \ldots, I = \lfloor \log n^{2/(2+d_o)} \rfloor$. Then $\mathcal{B}$ is the union of all these balls of different sizes, blown up by a factor of 4:

$$\mathcal{B} = \bigcup_{i=0}^{I} \{4B : B \in \mathcal{B}_i\}.$$

The partition $\mathbb{C}'$ is created by replacing each cell $C \in \mathbb{C}$ by two cells C'_1, C'_2 as follows:

- If $C \cap S = \emptyset$, then $C'_1 = C$ and $C'_2 = \emptyset$.
- Otherwise, let $i = \min\{I, \lceil \log(\Delta_{\mathcal{X}}/\Delta_n(C)) \rceil\}$; we'll find a ball $B \in \mathcal{B}_i$ such that $C \cap S \subset 4B$. One way to achieve this is by picking $x \in C \cap S$ and then choosing the ball $B \in \mathcal{B}_i$ whose center z is closest to x. Then define $C'_1 = C \cap 4B$ and $C'_2 = C \setminus C'_1$.

$\mathbb{C}'$ is the collection of all such C'_1, C'_2, over $C \in \mathbb{C}$. What makes this refined partition valuable is that the average physical diameter of its cells can be upper-bounded by the empirical data diameters of cells in $\mathbb{C}$, as captured in the following lemma. The last term represents the resolution of the balls in $\mathcal{B}$.

Lemma 33 *Let $\mathbb{C}$ be a partition of $\mathcal{X}$ and define $\mathbb{C}'$ as above. Then*

$$\sum_{C' \in \mathbb{C}'} \mu_n(C')\Delta^2(C') \leq 64\Delta_n^2(\mathbb{C}) + 256n^{-4/(2+d_o)}\Delta_{\mathcal{X}}^2.$$

3.5.4 Bounding the empirical masses of cells

Next, we need to relate the empirical masses $\mu_n(C)$ to their true values $\mu(C)$. Since the tree has height $O(\log n)$, each cell is the intersection of $O(\log n)$ half-spaces. The class of such convex sets has Vapnik-Chervonenkis dimension $O(d \log n)$, and thus a uniform convergence bound can easily be given for the empirical masses of these sets. However, this bound would depend linearly on the ambient dimension d, and we would like to avoid any such dependence.

Instead, we exploit the fact that the hyperplanes used for splitting are chosen at random, rather than by looking at the data. We then get the following bound.

Lemma 34 *There is a constant A_0 such that the following holds. Pick any $0 < \delta < 1$. With probability at least $1-\delta$ over the choice of S and the randomness in the algorithm, we have that for any partition $\mathbb{C} = \mathbb{C}^i$ generated during the construction of the tree, every cell $C' \in \mathbb{C}'$ satisfies*

$$\mu(C') \;\leq\; \mu_n(C') + 2\sqrt{\mu_n(C')\frac{V+\ln(4/\delta)}{n}} + 4\frac{V+\ln(4/\delta)}{n},$$

where $V \leq (A_0 \log n)(\log n + \text{loglog}(1/\delta))$.

3.5.5 Bounding the integrated excess risk

Recall that the algorithm starts with a partition $\mathbb{C}_0$ that has a single cell containing all the data, and then produces increasingly finer partitions $\mathbb{C}_1, \mathbb{C}_2, \ldots$, with data diameters shrinking by at least a factor of two. Recall also that the diameter decrease rate, denoted k, is defined to be the maximum increase in tree depth during each of these individual growth spurts.

For any $\mathbb{C}_i$, Lemmas 33 and 34 enable us to bound the expressions for variance and bias from Lemmas 31 and 32, whereupon we get the following risk bound.

Lemma 35 *There exists a constant A_1 independent of d and μ such that the following holds. Define $\alpha(n) \doteq \left(\log^2 n\right)\text{loglog}(1/\delta) + \log(1/\delta)$. With probability at least $1-\delta$ over the sample and the randomness in the algorithm (of Figure 10), for all partitions $\mathbb{C} = \mathbb{C}_i$ obtained,*

$$\|f_{n,\mathbb{C}} - f\|^2 \quad\leq\quad A_1\left(\Delta_{\mathcal{Y}}^2|\mathbb{C}|\frac{\alpha(n)}{n} + \lambda^2\left(\Delta_n^2(\mathbb{C}) + n^{-4/(2+d_o)}\Delta_{\mathcal{X}}^2\right)\right).$$

The algorithm chooses one particular partition $\mathbb{C}_i$ on which to base its regressor. To get some insight into the tradeoff involved, pretend for a moment that $\Delta_{\mathcal{X}}$, $\Delta_{\mathcal{Y}}$, and λ are all 1. Consider a partition $\mathbb{C}$ induced by the tree. If $\Delta_n(\mathbb{C}) = \zeta$, we would expect that the data diameter has been halved roughly $\log(1/\zeta)$ times. Since each halving grows the tree by $\leq k$ levels, $\mathbb{C}$ has depth at most $k\log(1/\zeta)$ in the tree, implying also that $|\mathbb{C}| \leq (1/\zeta)^k$. Plugging these values into the bound of Lemma 35, we get $\|f_{n,\mathbb{C}} - f\|^2 \lesssim \zeta^{-k}/n + \zeta^2$. Setting $\zeta = n^{-1/(2+k)}$ then gives the optimal bound $\|f_{n,\mathbb{C}^*} - f\|^2 \lesssim n^{-2/(2+k)}$. It can be shown that the automatic stopping rule gives a good approximation to this bound, from which Theorem 30 follows.

3.5.6 Related work

There have been a variety of different results on relating the behavior of nonparametric estimators to notions of intrinsic dimensionality.

Bickel and Li (2007) have shown that local kernel regressors are adaptive to manifold structure. Specifically, they obtain a bandwidth setting under which the asymptotic risk at any given point in $\mathbb{R}^d$ depends only on the manifold dimension and on the behavior of the kernel in the vicinity of that point. The appropriate bandwidth can be found either by estimating the manifold dimension or by cross-validating over possible values of this dimension.

Earlier work of Kulkarni and Posner (1995), although not treating the topic of adaptivity to intrinsic dimension, expresses the risk of 1-nearest neighbor regression in terms of the *box dimension* (Cutler, 1993; Clarkson, 2005) of the data, which is related to the doubling dimension. More recently, Kpotufe (2011) has shown that the risk of k-NN regression can be expressed in terms of intrinsic dimension.

A disadvantage of kernel and nearest neighbor regressors is that they are expensive to evaluate on a new data point. On a data set of size n, either kernel weights must be computed everywhere, resulting in an $\Omega(n)$ evaluation time, or the k_n nearest neighbors of a query point must be located, where k_n is optimally chosen as a root of n (Gyorfi et al., 2002). This sort of time complexity can be a burden in practice considering that nonparametric regression usually depends upon large data sizes for accuracy. Hence the appeal of an tree-based regressor that can be evaluated in $O(\log n)$ time. Methods for combining kernel and tree-based

regressors have been investigated by Kpotufe (2009), and are promising in that they are shown to yield both fast evaluation time as well as a good convergence rate of $n^{-2/(2+d_o)}$ for doubling dimension d_o (as opposed to the rate for RP trees, where the d_o term becomes $O(d_o \log d_o)$).

For classification problems, Scott and Nowak (2006) have shown that dyadic trees—whch split along coordinate directions, at the midpoint of a cell rather than the median—achieve convergence rates depending only on (something like) the box dimension, under smoothness conditions on the input distribution and the Bayes decision boundary. It can be shown that risk of a regressor based on dyadic partitioning does depend on d, but that this dependence appears in a leading constant (exponential in d) rather than in the exponent of n (Kpotufe and Dasgupta, 2012).

The random regression graph of Caponnetto and Smale (2007) is similar in spirit to a random projection tree since it also partitions space using random hyperplanes. However, its regression risk has only been analyzed in terms of a quantity that is different from the kinds of intrinsic dimension we have discussed: the norm of the regression function in the reproducing kernel Hilbert space induced by a specific kernel. In particular, this depends not just upon the predictor X but also on the response Y.

3.5.7 Some open problems

The diameter decrease rate for random projection trees, for data of doubling dimension d_o, has been shown to be $O(d_o \log d_o)$. It is unclear whether this can be improved to $O(d_o)$.

The notion of doubling dimension is applicable to any metric space. It would be useful to have a tree structure for general metric spaces that provably exhibits diameter decrease rates depending only on this dimension. Various metric trees have been proposed in the context of nearest neighbor search—for instance, the *cover tree* of Beygelzimer et al. (2006)—but their diameter decrease rates have not been determined.

It is also of great interest to move beyond doubling dimension to more general characterizations of the "degrees of freedom" of a distribution (or its support).

4 Active learning

Active learning refers to situations in which the ultimate goal is to find a good classifier or regressor $h : \mathcal{X} \to \mathcal{Y}$, as usual, but with an interesting twist on the learning process: unlabeled samples from $\mathcal{X}$ are available for free, while each corresponding label or response value must be explicitly queried and comes at a cost. The idea is to get the most out of a limited budget by choosing queries in an intelligent and adaptive manner.

In this article, we will largely restrict attention to classification problems in which $\mathcal{Y}$ is a finite set and the function h is to be selected from a pre-specified hypothesis class $\mathcal{H}$. Some of the ideas carry over to regression, but the connections have not yet been worked out thoroughly.

The backdrop to the theoretical developments we will describe is a practical landscape in which the increasing size of data sets has been making active learning ever more important. As a result, there has been a proliferation of querying heuristics. These differ in detail, but very often conform to the basic paradigm shown in Figure 12, which has a ready and intuitive appeal. We will begin by analyzing this schema within the usual framework of statistical learning theory, where we posit an unknown, underlying distribution $\mathbb{P}$ on $\mathcal{X} \times \mathcal{Y}$ from which data points (and their hidden labels) are drawn independently at random.

Any classifier $h \in \mathcal{H}$ is evaluated in terms of its performance on $\mathbb{P}$. The best candidate, $h^* \in \mathcal{H}$, is by definition the one with smallest error on $\mathbb{P}$, that is, with smallest

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

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

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

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,
  according to various criteria)

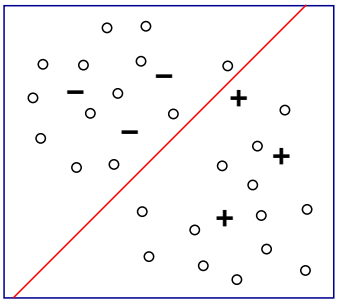


Figure 12: *Left:* A basic framework to which many active learning heuristics conform. *Right:* Here the current boundary is a linear separator, and according to the basic schema, there are several unlabeled points close to it that would be candidates for querying.

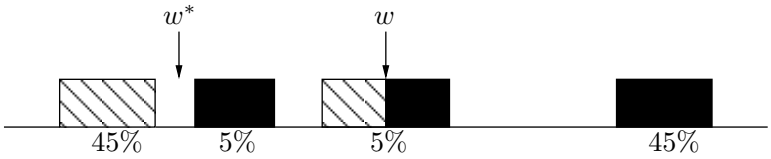


Figure 13: An illustration of sampling bias in active learning. The data lie in four groups on the line, and are (say) distributed uniformly within each group. The two extremal groups contain 90% of the distribution. Solids have a + label, while stripes have a − label.

especially amenable to analysis. The algorithms we describe, however, are designed for the more realistic *nonseparable* scenario.

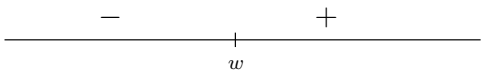
4.1 Sampling bias

When we consider the querying scheme of Figure 12 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 (obtained before the main querying loop begins), 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 13. 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

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



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. For a discussion of this "hidden cluster" problem in text classification, see Schutze et al. (2006).

Sampling bias is the most fundamental challenge posed by active learning. Three broad strategies have been developed for managing it:

1. Label everything.
2. Use importance weights.
3. Explicitly manage sampling regions.

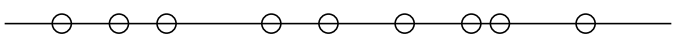
We will look at each of these, but first we pause to consider whether active learning is even worth the trouble: how much benefit can we possibly hope to get from intelligent querying? Can the number of labels queried, the so-called *label complexity*, really be much lower than in supervised learning, where the query points are essentially chosen at random?

4.2 How much can active learning help?

When the data lie on the real line and the hypotheses $\mathcal{H}$ are thresholds, how many labels are needed to find $h \in \mathcal{H}$ whose error on the underlying distribution $\mathbb{P}$ is at most some ϵ?

In supervised learning, such issues are well understood. The standard machinery of sample complexity (Bousquet et al., 2004) 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 by a binary search that asks for just $\log 1/\epsilon$ labels: first ask for the label of the median point; if it is $+$, 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 reduction 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 $\log 1{,}000{,}000 \approx 20$.

This toy example is only for separable data, but with a little care something similar can be achieved for the nonseparable case. As we will now see, this even holds for more complicated hypothesis classes $\mathcal{H}$, in many situations.

We turn to the first method for managing sampling bias: labeling everything.

4.3 Managing bias by labeling everything

The paper of Cohn et al. (1994) introduced a simple active learning strategy for separable data. This scheme, henceforth nicknamed CAL after its authors, 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.

CAL works by 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 queried (Figure 14, left). The following observation needs no proof.

Lemma 36 *Suppose the CAL algorithm is given a stream of data that is separable by some $h^* \in \mathcal{H}$. Then at any time t, the version space $\mathcal{H}_t$ consists of exactly those $h \in \mathcal{H}$ that agree with h^* on the labels of $x_1, \ldots, x_{t-1}$.*

In particular, this means that at any time t, CAL can perform as well as a supervised learner that has seen t labeled examples. The interesting question is how many of the t instances CAL queries.

Cohn-Atlas-Ladner algorithm:
$\mathcal{H}_1 = \mathcal{H}$
For $t = 1, 2, \ldots$:
  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:
    $\mathcal{H}_{t+1} = \mathcal{H}_t$

Implicit version:
$S = \{\}$ (points seen so far)
For $t = 1, 2, \ldots$:
  Receive unlabeled point x_t
  If `learn`$(S \cup \{(x_t, +1)\})$ and `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 14: *Left:* CAL, a generic active learner for separable data. *Right:* A way to simulate CAL without having to explicitly maintain the version space $\mathcal{H}_t$. Here `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.

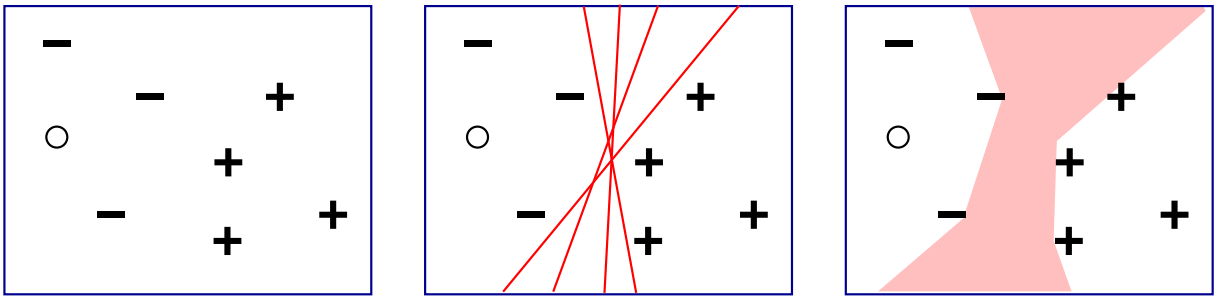


Figure 15: *Left:* The first seven points in the data stream were labeled. How about this next point, shown by a circle? *Middle:* Some of the hypotheses in the current version space. *Right:* The region of disagreement.

To get a feel for this, consider Figure 15, which 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 in a more general setting, there is a broad class of situations in which this $\mathbb{P}$-mass halves every constant number of labels, giving a label complexity that is exponentially smaller 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.

4.3.1 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. A succession of papers, starting with Balcan et al. (2006) and Dasgupta et al. (2007), have shown how to overcome these hurdles.

The first problem can be handled by maintaining the version space implicitly in terms of the labeled examples seen so far (Figure 14, right). For the second problem, nonseparable data, we need to change the definition of the version space $\mathcal{H}_t$, because there might not exist any hypotheses that agree with all the labels. Thereafter, with the newly-defined $\mathcal{H}_t$, it is possible to 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, the label of x_t can be "inferred"—call this $\hat{y}_t$—and included 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

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

Figure 16: The DHM selective sampling algorithm. Here, $\text{err}(h, A)$ refers to the empirical error $(1/|A|)\sum_{(x,y)\in A} 1(h(x) \neq y)$. A possible setting for Δ_t is shown in Equation (4). At any time, the current hypothesis is $\texttt{learn}(I, Q)$.

resulting algorithm, which we will call DHM after its authors (Dasgupta et al., 2007), is shown in Figure 16.

The DHM algorithm makes black-box calls to a special type of supervised learner: for $A, B \subset \mathcal{X} \times \{\pm 1\}$,

> $\texttt{learn}(A, B)$ 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 computational hardness of minimizing the 0–1 loss on B; we will return to this issue later.

During the learning process, every point gets labeled: this label is either inferred, in which case the point is put in set I, or it is queried, in which case the point is in set Q. Since no point is discarded, there is no bias introduced into the marginal distribution on $\mathcal{X}$. However, there is a danger of having bias in the conditional distribution of y given x, because the inferred labels I can differ from the actual labels—call them $\widetilde{I}$. We now discuss how this is controlled.

Standard generalization theory, based on VC dimension (Bousquet et al., 2004), gives uniform bounds on how much the empirical error estimates based on $\widetilde{I} \cup Q$ differ from error rates on the underlying distribution $\mathbb{P}$, that is, on the quantity

$$\sup_{h\in\mathcal{H}} |\text{err}(h, \widetilde{I} \cup Q) - \text{err}(h, \mathbb{P})|.$$

This is not helpful for us because we only have the inferred labels I, not the true labels $\widetilde{I}$. However, similar bounds apply to differences between pairs of classifiers,

$$\sup_{h,h'\in\mathcal{H}} (\text{err}(h, \widetilde{I} \cup Q) - \text{err}(h', \widetilde{I} \cup Q)) - (\text{err}(h, \mathbb{P}) - \text{err}(h', \mathbb{P})).$$

This is more useful because all the hypotheses h, h' we consider are consistent with I, and hence any discrepancies between I and $\widetilde{I}$ translate their error estimates equally:

$$\text{err}(h, I \cup Q) - \text{err}(h', I \cup Q) = \text{err}(h, \widetilde{I} \cup Q) - \text{err}(h', \widetilde{I} \cup Q).$$

Thus, even though the algorithm uses I in place of $\widetilde{I}$, the relative ordering of hypotheses is preserved.

The generalization bound Δ_t needed in the DHM algorithm can be set to

$$\begin{aligned} \Delta_t &= \beta_t^2 + \beta_t \left(\sqrt{\text{err}(h_{+1}, I_t \cup Q_t)} + \sqrt{\text{err}(h_{-1}, I_t \cup Q_t)} \right) \\ \beta_t &= C\sqrt{\frac{d \log t + \log(1/\delta)}{t}} \end{aligned} \tag{4}$$

where C is a universal constant, d is the VC dimension of class $\mathcal{H}$, δ is the overall permissible failure probability, and I_t, Q_t are the inferred and queried labels at time t.

Following the above argument, we can then assert that even though the inferred labels might not correspond to their hidden values, they are consistent with the optimal hypothesis. Further details can be found in the paper of Dasgupta et al. (2007).

Lemma 37 *Suppose there is some $h^* \in \mathcal{H}$ that achieves $err(h^*, \mathbb{P}) = \inf_{h \in \mathcal{H}} err(h, \mathbb{P})$. Then with probability at least $1 - \delta$ over the sample, all labels inferred by the DHM algorithm agree with h^*.*

(If there isn't an h^*, a similar statement can be made asserting that forcing agreement with I_t doesn't increase the optimal error rate.)

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 summarized in Figure 12? To understand this, let's return to the example of Figure 13. 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.

4.3.2 Label complexity

The label complexity of supervised learning is quite well characterized by the Vapnik-Chervonenkis (VC) dimension of the concept class (Bousquet et al., 2004): 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 (Dasgupta, 2005). The work of Hanneke (2007) 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. We will shortly define this parameter and see examples of it, but in the meanwhile, we take a look at the bounds.

Suppose that either CAL or DHM is supplied with data generated independently at random from an underlying distribution $\mathbb{P}$. At any given time, either algorithm can be made to generate a hypothesis $h \in \mathcal{H}$ by a suitable call to its `learn` black box. We say the algorithm has label complexity $\mathcal{L} : (0,1) \times (0,1) \to \mathbb{N}$, if for any $0 < \epsilon, \delta < 1$, with probability at least $1 - \delta$, at any time after it makes $\mathcal{L}(\epsilon, \delta)$ queries, it returns a hypothesis with error at most $\inf_{h \in \mathcal{H}} \mathrm{err}(h) + \epsilon$.

In typical supervised learning bounds, and here as well, the dependence of $\mathcal{L}(\epsilon, \delta)$ upon δ is modest, at most $\mathrm{poly}\log(1/\delta)$. To avoid clutter, we will henceforth ignore δ and speak only of $\mathcal{L}(\epsilon) = \mathcal{L}(\epsilon, 1/4)$, where the constant $1/4$ has been chosen arbitrarily.

Theorem 38 *(Hanneke, 2011) Suppose $\mathcal{H}$ has finite VC dimension d, and the learning problem is separable, with disagreement coefficient θ. Then CAL achieves a label complexity of*

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

where the $\widetilde{O}$ 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 $1/\epsilon$.

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

Theorem 39 *(Dasgupta et al., 2007) With parameters as defined above, and $\nu = \inf_{h\in\mathcal{H}} err(h)$, DHM achieves label complexity*

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

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 40 *(Beygelzimer et al., 2009) 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 $\nu = \inf_{h\in\mathcal{H}} err(h)$.

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

4.3.3 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 17 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

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

This quantity has also been studied earlier in empirical process theory, under the name of *Alexander capacity function* (Gine and Koltchinskii, 2006).

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:

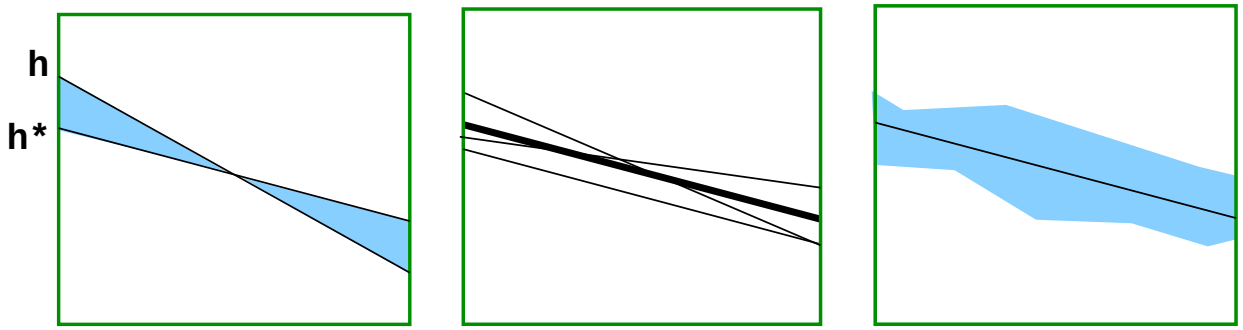


Figure 17: *Left:* Suppose the data lie in the plane, and that hypothesis class consists of linear separators. The distance between any two hypotheses h^* and h is the probability mass (under $\mathbb{P}$) of the region on which they disagree. *Middle:* The thick line is h^*. The thinner lines are examples of hypotheses in $B(h^*, r)$. *Right:* $\mathrm{DIS}(B(h^*, r))$ might look something like this.

- 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 (Hanneke, 2007): $\theta \leq \sqrt{d}$.
- Linear separators in $\mathbb{R}^d$, with a smooth data density bounded away from zero (Friedman, 2009): $\theta = c(h^*)d$, where $c(h^*)$ is some constant depending on the target hypothesis h^*.

We now give some informal intuition for the label complexity of CAL (Theorem 38); the argument for DHM is similar. Recall that CAL deals with the separable case, in which some $h^* \in \mathcal{H}$ has zero error. At any time t, the algorithm has t points drawn at random from $\mathbb{P}$, and has labels for them that are consistent with h^*. By standard generalization bounds, with high probability, any hypothesis $h \in \mathcal{H}$ consistent with these t examples has error at most $\Delta_t = O(d/t)$, ignoring some logarithmic terms, where d is the VC dimension of $\mathcal{H}$. Thus the version space $\mathcal{H}_t$ lies within $B(h^*, \Delta_t)$, and, by the definition of disagreement coefficient,

$$\mathbb{P}(\mathrm{DIS}(\mathcal{H}_t)) \;\leq\; \mathbb{P}(\mathrm{DIS}(B(h^*, \Delta_t))) \;\leq\; \theta\Delta_t.$$

Thus, the next data point to arrive has at most this probability of being queried.

Putting these various facts together, the error rate Δ_t drops below ϵ at time $T = O(d/\epsilon)$, and the expected number of queries made up to that point is, roughly,

$$\sum_{t=1}^{T} \theta\Delta_t = O\left(\theta d \log T\right).$$

4.3.4 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 symmetric difference. 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 (Balcan et al., 2008; Friedman, 2009) analyzes such cases by loosening the definition of disagreement coefficient from

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

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

4.4 Managing bias using importance weights

The methods of the previous section are built explicitly for 0–1 loss and are not easily adapted to other loss functions. This is problematic for applications in which other losses would be preferable. Even more crucially, minimizing the empirical 0–1 loss is known to be computationally intractable for many hypothesis classes of interest such as linear separators (Feldman et al., 2009); in such cases, the `learn` subroutine required by DHM would also be inefficient in general.

We now consider an alternative approach to active learning that is geared towards general loss functions, including in particular the convex loss functions for which empirical loss minimization is computationally tractable. Most of this section is taken from the paper of Beygelzimer et al. (2009); the algorithm and bounds were further refined by Beygelzimer et al. (2010).

As before, let $\mathcal{X}$ denote the instance space and $\mathcal{Y}$ the label space. Let $\mathcal{H}$ be a hypothesis class consisting of functions $h : \mathcal{X} \to \mathcal{Z}$, where $\mathcal{Z}$ is some prediction space. Any hypothesis $h \in \mathcal{H}$ is evaluated according to a loss function $\ell : \mathcal{Z} \times \mathcal{Y} \to \mathbb{R}^+$. When $\mathcal{Y} = \{+1, -1\}$, the 0–1 loss takes $\mathcal{Z} = \mathcal{Y}$ and $\ell(z, y) = \mathbf{1}(yz < 0)$. There are also many loss functions that allow $\mathcal{Z} = \mathbb{R}$, for instance:

- Hinge loss: $\ell(z, y) = (1 - yz)_+$
- Logistic loss: $\ell(z, y) = \ln(1 + e^{-yz})$
- Squared loss: $\ell(z, y) = (1 - yz)^2$
- Absolute loss: $\ell(z, y) = |1 - yz|$

4.4.1 Importance-weighted active learning

Unlike the algorithms of the previous section, importance-weighted active learning (henceforth, IWAL) does not attempt to label points it hasn't queried. However, it assigns each query point an importance weight that is crucial in evaluating hypotheses. Thus, its data set S_t at any time t consists of triples (x, y, w) where x is an instance, y is its queried label, and w is the weight. The empirical loss of any hypothesis $h \in \mathcal{H}$ is taken to be

$$L_t(h) = \frac{1}{|S_t|} \sum_{(x,y,w) \in S_t} w\, \ell(h(x), y).$$

This is constructed so as to be an unbiased estimate of the true loss

$$L(h) = \mathbb{E}_{(X,Y) \sim \mathbb{P}} \ell(h(X), Y).$$

$S_0 = \emptyset$ (labeled data set)
For $t = 1, 2, \ldots$:
  Receive x_t
  Choose $p_t \in [0, 1]$ based on everything seen so far
  Flip a coin $Q_t \in \{0, 1\}$ with bias p_t
  If $Q_t = 1$:
    Request y_t and set $S_t = S_{t-1} \cup \{(x_t, y_t, 1/p_t)\}$
  Else:
    $S_t = S_{t-1}$

Figure 18: Importance-weighted active learning. The hypothesis that would be chosen at time t is $h_t = \arg\min_{h \in H} \sum_{(x,y,c) \in S_t} c \cdot \ell(h(x), y)$.

The overall framework is shown in Figure 18. The learner sees a stream of unlabeled points x_t and for each one, decides on the spot whether or not to query it. However, this decision is not deterministic. Instead, the learner picks a number $p_t \in [0, 1]$ based on everything it has seen up to that point, and queries y_t with this probability. If y_t is queried, the importance weight of (x_t, y_t) is taken to be $1/p_t$.

It is not hard to see that in this setup, $\mathbb{E}[L_t(h)] = L(h)$. The following uniform convergence bound also follows easily if the probabilities p_t are bounded below.

Lemma 41 *Suppose $\mathcal{H}$ is a finite hypothesis class, and that the loss function ℓ takes values in the interval $[0, 1]$. Pick any $0 < \delta < 1$ and any time T. If there is a constant p_o such that $p_t \geq p_o$ for all $1 \leq t \leq T$, then*

$$\Pr\left[\max_{h \in H} |L_T(h) - L(h)| > \sqrt{\frac{2}{p_o^2}\left(\frac{\ln|\mathcal{H}| + \ln(2/\delta)}{T}\right)}\right] < \delta.$$

PROOF: For any $h \in H$, consider the sequence of random variables $U_1, \ldots, U_T$:

$$U_t = \frac{Q_t}{p_t}\ell(h(x_t), y_t) - L(h).$$

Then $Z_t = \sum_{i=1}^{t} U_i$ is a martingale, letting $Z_0 = 0$. Indeed, for any $1 \leq t \leq T$,

$$\begin{aligned}\mathbb{E}[Z_t \mid Z_{t-1}, ..., Z_0] &= \mathbb{E}\left[U_t + Z_{t-1} \mid Z_{t-1}, ..., Z_0\right] \\ &= \mathbb{E}\left[\,\ell(h(x_t), y_t) - L(h) + Z_{t-1} \mid Z_{t-1}, ..., Z_0\,\right] \;=\; Z_{t-1}.\end{aligned}$$

Observe that $|Z_{t+1} - Z_t| = |U_{t+1}| \leq 1/p_o$ for all $0 \leq t < T$. Using $Z_T = T(L_T(h) - L(h))$ and applying Azuma's inequality, we have that for any $\lambda > 0$,

$$\Pr\left[|L_T(h) - L(h)| > \frac{\lambda}{p_o\sqrt{T}}\right] < 2e^{-\lambda^2/2}.$$

The result follows by setting $\lambda = \sqrt{2(\ln|\mathcal{H}| + \ln(2/\delta))}$ and taking a union bound over $h \in \mathcal{H}$. □

4.4.2 Choosing the sampling probabilities

The paper of Beygelzimer et al. (2010) describes a method for setting the sampling probabilities p_t that is similar in flavor to DHM:

- Define the version space at time t by:

$$\mathcal{H}_0 = \mathcal{H}$$
$$\mathcal{H}_t = \{h \in \mathcal{H}_{t-1} : L_{t-1}(h) \leq L^*_{t-1} + \Delta_{t-1}\}$$

  where $L^*_{t-1} = \inf_{h \in \mathcal{H}_{t-1}} L_{t-1}(h)$, and Δ_t is a standard generalization bound.

- Set the sampling probability for a new point x_t to be

$$p_t = \min\left(1, \sup_{f,g \in \mathcal{H}_t, y \in \mathcal{Y}} \ell(f(x_t), y) - \ell(g(x_t), y)\right).$$

For convex loss functions ℓ, this computation can be managed efficiently. Even though the resulting probabilities are not bounded below—and thus Lemma 41 does not apply—it can be shown that at any time t, any hypothesis h in the version space $\mathcal{H}_t$ satisfies $L(h) - L^* \leq 2\Delta_{t-1}$, where $L^* = \inf_{h \in \mathcal{H}} L(h)$.

The original paper describes a generalization of the disagreement coefficient to general loss functions, and uses it to give label complexity bounds for IWAL. Various lower bounds for active learning under loss functions have also been obtained: see Hanneke and Yang (2010).

In summary, importance weighting helps make the ideas of the previous section more practical by bringing them into the framework of convex loss minimization. A particularly efficient realization of this scheme, suitable for large-scale learning, has been described by Karampatziakis and Langford (2011); however, the label complexity of that method has not been characterized.

4.5 Managing bias by explicitly controlling sampling regions

Here is a natural active learning strategy: start with a pool of unlabeled data, cluster it, ask for one label per cluster, and then take these labels as representative of their entire respective clusters. Such a scheme has an intuitive appeal but 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 et al. (2003). Their algorithm first imposes a neighborhood graph on an unlabeled data set S, and then locally propagates any labels it obtains, roughly as follows:

1. Pick a few points from S at random and get their labels.
2. Repeat:
   - Propagate labels to "nearby" unlabeled points in the graph
   - Query in an "unknown" part of the graph

This kind of nonparametric learner would likely have the usual problems of sampling bias, but is different from the approaches of the previous section, and has been studied far less. One scheme due to Dasgupta and Hsu (2008), which we will call HS (for *hierarchical sampling*), 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.

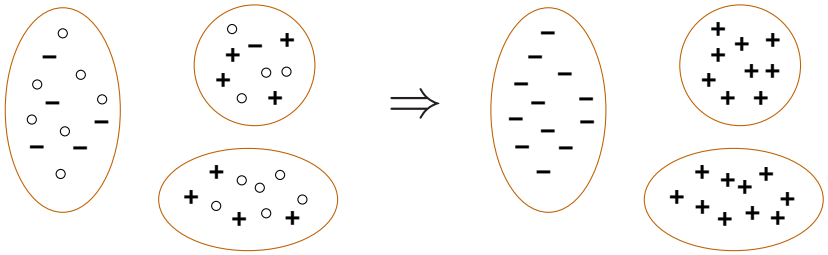
4.5.1 Three rules for querying

The HS scheme starts with a pool of unlabeled data $S \subset \mathcal{X}$. It queries some of these points and ultimately produces a labeling of the entire set S that is guaranteed to be correct on at least a $1-\epsilon$ fraction of the points, with probability at least $1-\delta$ (over its random choices), where ϵ and δ are user-specified accuracy and confidence parameters. The algorithm always works with a partition, or clustering, of S. This can initially be constructed in any manner, and is allowed to change over time. Let $\mathbb{C}_t$ be the partition operative at time t, when the tth query is being chosen. To avoid sampling bias, any querying must conform to a simple rule:

> **Rule 1.** At any time t, the learner specifies a cluster $C \in \mathbb{C}_t$ and receives a point chosen at random from $S \cap C$, along with its label.

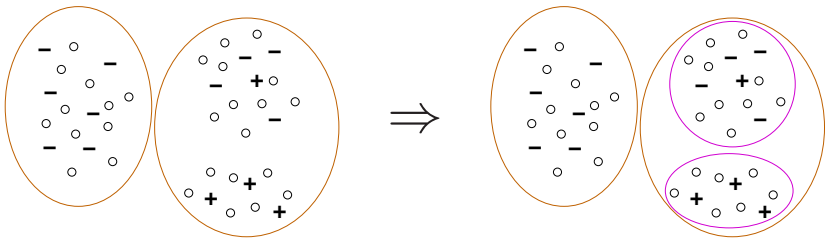
In other words, the learner is not allowed to specify exactly which point to label, but merely the cluster from which it will be randomly drawn. In the case where there are binary labels $\{-1, +1\}$, it helps to think of each cluster as a biased coin whose heads probability is simply the fraction of 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 partition $\mathbb{C}_T$ 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.

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



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

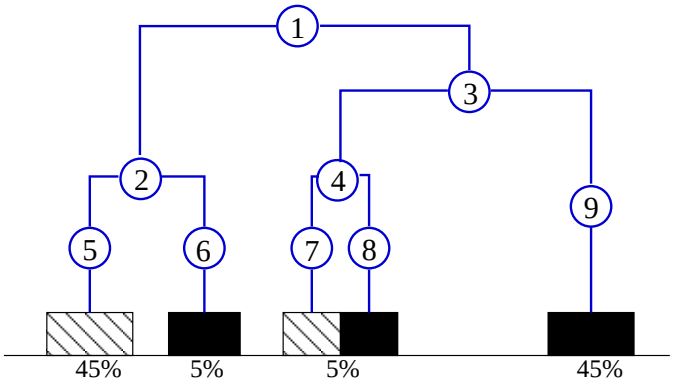


Figure 19: The top few nodes of a hierarchical clustering.

As in the example above, the nested structure of clusters makes it possible to re-use labels when the partition 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 partition is refined.

> **Rule 3.** When a cluster is split to obtain a new partition, $\mathbb{C}_t \to \mathbb{C}_{t+1}$, 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.

4.5.2 An illustrative example

Given the three rules for querying, the benefits of the HS framework are quite intuitive. We can get a sense of its behavior by revisiting the example in Figure 13 that presented difficulties for many active learning heuristics.

HS starts with a hierarchical clustering of this data. Figure 19 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 will reveal that the top node 1 is very mixed, whereupon the partition $\{1\}$ will be replaced by $\{2, 3\}$. Subsequent random samples will be chosen from either 2 or 3, according to a sampling strategy that is arbitrary but may, for instance, favor the less-pure node. A few more queries down the line, the pruning will likely be refined to $\{2, 4, 9\}$. This is when the benefits of the partitioning scheme will become most obvious; based on the samples seen, it will be concluded that cluster 9 is (almost) pure, and thus (almost) no more queries need to be 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, HS 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.

```
Fix a random ordering of the points S
Build tree T (such as k-d tree or RP tree) from S
ℓ = 0   // (current level in tree)
ℂ = {root of T}   // (currently active cells)
While ℂ ≠ ∅:
    Set m_ℓ = ((2ℓ ln 2) + ln(1/δ))/ε
    For each cell C ∈ ℂ:
        Remove C from ℂ
        Pick the first m_ℓ points in C ∩ S, and query their labels
        (some might have been obtained earlier)
        If all these labels are the same:
            Label all points in C ∩ S with this label
        Else:
            If there are unqueried points in C ∩ S, add the children of C (in T) to ℂ
    Increment level ℓ
Return the fully-labeled sample S
```

Figure 20: An instantiation of the HS scheme, starting with a set $S \subset \mathcal{X}$ of unlabeled points, and user-specified accuracy and confidence parameters $0 < \epsilon, \delta < 1$.

4.5.3 A specific instantiation of the HS framework

So far we have merely described a framework for hierarchical sampling that avoids sampling bias. A specific realization of this framework would need to specify:

- The manner in which the initial hierarchical clustering is built.
- The rule for deciding which cluster to query.
- The criterion for deciding when to move down from a cluster to its two children.

The original paper of Dasgupta and Hsu (2008) contains one scheme of this sort. We now discuss an alternative proposed by Urner et al. (2013). To keep things as simple as possible, we assume $|\mathcal{Y}| = 2$, although the method generalizes immediately to any finite label set.

The algorithm is shown in Figure 20. It starts with a large pool of unlabeled points S, and builds a hierarchical clustering on them. Initially, the operative clustering $\mathbb{C}$ consists of a single cluster, the root cell. As querying proceeds, $\mathbb{C}$ gradually moves down the tree: on the ℓth iteration, it is a subset of the cells in level ℓ of the tree. For each of these cells, $O(\ell/\epsilon)$ labels are obtained—some of which will have carried over from previous levels—and if they are all identical, then the cell is declared pure and all of the data points falling in it are labeled accordingly. For the non-pure cells, their children are part of $\mathbb{C}$ in the next level.

The labels obtained in a cell provide an unbiased estimate of the overall balance of labels within that cell. The sampling error can be bounded using Chernoff-Hoeffding bounds, and by taking a union bound over all cells, we obtain the following basic guarantee.

Lemma 42 *With probability at least $1 - \delta$, the final labeling output by the algorithm of Figure 20 is correct on at least a $1 - \epsilon$ fraction of the points.*

This scheme will query just a small fraction of the points S if the hierarchical clustering T has large clusters, near the root of the tree, that are pure in their labels. Urner et al. (2013) consider a formal model in which:

- $\mathcal{Y} = \{0, 1\}$,
- the regression function $\eta(x) = \mathbb{E}[Y|X = x]$ takes values in $\{0, 1\}$, and

- η satisfies a relaxed Lipschitz property.

Under these conditions, they give label complexity bounds that depend on the Lipschitz constant and the data diameter decrease rate (along the lines of Definition 29) of the tree T.

It is an interesting open problem to develop hierarchical sampling strategies whose label complexity can be analyzed in more general situations.

Acknowledgments

During the writing of this article, the author was supported by the National Science Foundation under grant IIS-1162581.

References

Achlioptas, D. and McSherry, F. (2005). On spectral learning of mixtures of distributions. In *Proceedings of the 18th Annual Conference on Learning Theory*, pages 458–469.

Amenta, N. and Bern, M. (1999). Surface reconstruction by voronoi filtering. *Discrete and Computational Geometry*, 22:481–504.

Assouad, P. (1983). Plongements lipschitziens dans $\mathbb{R}^n$. *Bull. Soc. Math. France*, 111(4):429–448.

Balcan, M.-F., Beygelzimer, A., and Langford, J. (2006). Agnostic active learning. In *Proceedings of the 23rd International Conference on Machine Learning*.

Balcan, M.-F., Hanneke, S., and Wortman, J. (2008). The true sample complexity of active learning. In *Proceedings of the 21st Annual Conference on Learning Theory*.

Baraniuk, R. and Wakin, M. (2009). Random projections of smooth manifolds. *Foundations of Computational Mathematics*, 9(1):51–77.

Belkin, M. and Niyogi, P. (2003). Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15:1373–1396.

Belkin, M. and Sinha, K. (2010). Polynomial learning of distribution families. In *Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science*.

Beygelzimer, A., Dasgupta, S., and Langford, J. (2009). Importance weighted active learning. In *Proceedings of the 26th International Conference on Machine Learning*.

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

Beygelzimer, A., Kakade, S., and Langford, J. (2006). Cover trees for nearest neighbor. In *Proceedings of the 23rd International Conference on Machine Learning*.

Bickel, P. and Li, B. (2007). Local polynomial regression on unknown manifolds. *Complex Datasets and Inverse Problems: Tomography, Networks, and Beyond, IMS Lecture Notes – Monograph Series*, 54:177–186.

Bousquet, O., Boucheron, S., and Lugosi, G. (2004). Introduction to statistical learning theory. *Lecture Notes in Artificial Intelligence*, 3176:169–207.

Breiman, L., Friedman, J., Stone, C., and Olshen, R. (1984). *Classification and Regression Trees*. Chapman and Hall.

Brubaker, S. and Vempala, S. (2008). Isotropic PCA and affine-invariant clustering. In *Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science*, pages 551–560.

Caponnetto, A. and Smale, S. (2007). Risk bounds for random regression graphs. *Foundations of Computational Mathematics*, 7:495–528.

Chaudhuri, K. and Dasgupta, S. (2010). Rates of convergence for the cluster tree. In *Advances in Neural Information Processing Systems*.

Chaudhuri, K., Kakade, S., Livescu, K., and Sridharan, K. (2009). Multiview clustering via canonical correlation analysis. In *Proceedings of the 26th International Conference on Machine Learning*.

Chaudhuri, K. and Rao, S. (2008). Learning mixtures of product distributions using correlations and independence. In *Proceedings of the 21st Annual Conference on Learning Theory*.

Clarkson, K. (2005). Nearest-neighbor searching and metric space dimensions. In *Nearest-Neighbor Methods for Learning and Vision: Theory and Practice*. MIT Press.

Clarkson, K. (2008). Tighter bounds for random projections of manifolds. In *Proceedings of the 24th Annual Symposium on Computational Geometry*, pages 39–48.

Cohn, D., Atlas, L., and Ladner, R. (1994). Improving generalization with active learning. *Machine Learning*, 15(2):201–221.

Cutler, C. (1993). A review of the theory and estimation of fractal dimension. In Tong, H., editor, *Dimension Estimation and Models*, pages 1–107. World Scientific.

Dasgupta, S. (2005). Coarse sample complexity bounds for active learning. In *Advances in Neural Information Processing Systems*.

Dasgupta, S. (2008). The hardness of k-means clustering. *UCSD CSE Technical Report 2008-0916*.

Dasgupta, S. and Freund, Y. (2008). Random projection trees and low dimensional manifolds. In *ACM Symposium on Theory of Computing*, pages 537–546.

Dasgupta, S. and Hsu, D. (2008). Hierarchical sampling for active learning. In *Proceedings of the 25th International Conference on Machine Learning*.

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

Dasgupta, S. and Schulman, L. (2007). A probabilistic analysis of EM for mixtures of separated, spherical gaussians. *Journal of Machine Learning Research*, 8:203–226.

Dempster, A., Laird, N., and Rubin, D. (1977). Maximum-likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39:1–38.

Devroye, L., Gyorfi, L., and Lugosi, G. (1996). *A Probabilistic Theory of Pattern Recognition*. Springer.

Dudley, R. (1979). Balls in $\mathbb{R}^k$ do not cut all subsets of $k+2$ points. *Advances in Mathematics*, 31(3):306–308.

Feldman, V., Gopalan, P., Khot, S., and Ponnuswami, A. (2009). On agnostic learning of parities, monomials and halfspaces. *SIAM Journal on Computing*, 39(2):606–645.

Forgy, E. (1965). Cluster analysis of multivariate data: efficiency versus interpretability of classifications. *Biometrics*, 21:768–769.

Friedman, E. (2009). Active learning for smooth problems. In *Proceedings of the 22nd Annual Conference on Learning Theory*.

Garey, M. and Johnson, D. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman.

Gine, E. and Koltchinskii, V. (2006). Concentration inequalities and asymptotic results for ratio type empirical processes. *Annals of Probability*, 34(3):1143–1216.

Gupta, A., Krauthgamer, R., and Lee, J. R. (2003). Bounded geometries, fractals, and low-distortion embeddings. In *44th Annual IEEE Symposium on Foundations of Computer Science*, pages 534–543.

Gyorfi, L., Kohler, M., Krzyzak, A., and Walk, H. (2002). *A Distribution-free Theory of Nonparametric Regression*. Springer.

Hanneke, S. (2007). A bound on the label complexity of agnostic active learning. In *Proceedings of the 25th International Conference on Machine Learning*.

Hanneke, S. (2011). Rates of convergence in active learning. *Annals of Statistics*, 39(1):333–361.

Hanneke, S. and Yang, L. (2010). Negative results for active learning with convex losses. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*.

Hartigan, J. (1981). Consistency of single linkage for high-density clusters. *Journal of the American Statistical Association*, 76(374):388–394.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning*. Springer, 2nd edition.

Hsu, D. and Kakade, S. (2013). Learning mixtures of spherical Gaussians: moment methods and spectral decompositions. In *Fourth Innovations in Theoretical Computer Science*.

Indyk, P. and Naor, A. (2007). Nearest neighbor preserving embeddings. *ACM Transactions on Algorithms*, 3(3).

Johnson, W. and Lindenstrauss, J. (1984). Extensions of Lipschitz maps into a Hilbert space. *Contemporary Mathematics*, 26:189–206.

Kalai, A., Moitra, A., and Valiant, G. (2012). Disentangling Gaussians. *Communications of the ACM*, 55(2):113–120.

Kannan, R., Salmasian, H., and Vempala, S. (2008). The spectral method for general mixture models. *SIAM Journal on Computing*, 38(3):1141–1156.

Karampatziakis, N. and Langford, J. (2011). Importance weight aware gradient updates. In *Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence*.

Kpotufe, S. (2009). Fast, smooth and adaptive regression in metric spaces. In *Advances in Neural Information Processing Systems*.

Kpotufe, S. (2011). K-NN regression adapts to local intrinsic dimension. In *Advances in Neural Information Processing Systems*.

Kpotufe, S. and Dasgupta, S. (2012). A tree-based regressor that adapts to intrinsic dimension. *Journal of Computer and System Sciences*, 78(5):1496–1515.

Kpotufe, S. and von Luxburg, U. (2011). Pruning nearest neighbor cluster trees. In *Proceedings of the 28th International Conference on Machine Learning*.

Kruskal, J. (1956). On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7(1):48–50.

Kulkarni, S. and Posner, S. (1995). Rates of convergence of nearest neighbor estimation under arbitrary sampling. *IEEE Transactions on Information Theory*, 41(4):1028–1039.

Lloyd, S. (1982). Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137.

Mahajan, M., Nimbhorkar, P., and Varadarajan, K. (2012). The planar k-means problem is NP-hard. *Theoretical Computer Science*, 442:13–21.

Maier, M., Hein, M., and von Luxburg, U. (2009). Optimal construction of k-nearest neighbor graphs for identifying noisy clusters. *Theoretical Computer Science*, 410:1749–1764.

Mitchell, T. (1997). *Machine Learning*. McGraw-Hill.

Mohri, M., Rostamizadeh, A., and Talwalkar, A. (2012). *Foundations of Machine Learning*. MIT Press.

Niyogi, P., Smale, S., and Weinberger, S. (2006). Finding the homology of submanifolds with high confidence from random samples. *Discrete and Computational Geometry*.

Pearson, K. (1894). Contributions to the mathematical theory of evolution. *Philosophical Transactions of the Royal Society, Series A*, 185:71–110.

Penrose, M. (1995). Single linkage clustering and continuum percolation. *Journal of Multivariate Analysis*, 53:94–109.

Redner, R. and Walker, H. (1984). Mixture densities, maximum likelihood and the EM algorithm. *SIAM Review*, 26(2):195–239.

Rigollet, P. and Vert, R. (2009). Fast rates for plug-in estimators of density level sets. *Bernoulli*, 15(4):1154–1178.

Rinaldo, A., Singh, A., Nugent, R., and Wasserman, L. (2012). Stability of density-based clustering. *Journal of Machine Learning Research*, 13:905–948.

Rinaldo, A. and Wasserman, L. (2010). Generalized density clustering. *Annals of Statistics*, 38(5):2678–2722.

Roweis, S. and Saul, L. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290:2323–2326.

Schutze, H., Velipasaoglu, E., and Pedersen, J. (2006). Performance thresholding in practical text classification. In *ACM International Conference on Information and Knowledge Management*.

Scott, C. and Nowak, R. (2006). Minimax-optimal classification with dyadic decision trees. *IEEE Transactions on Information Theory*, 52(4):1335–1354.

Singh, A., Scott, C., and Nowak, R. (2009). Adaptive Hausdorff estimation of density level sets. *Annals of Statistics*, 37(5B):2760–2782.

Steinwart, I. (2011). Adaptive density level set clustering. In *Proceedings of the 24th Annual Conference on Learning Theory*.

Stone, C. (1980). Optimal rates of convergence for nonparametric estimators. *Annals of Statistics*, 8(6):1348–1360.

Stone, C. (1982). Optimal global rates of convergence for nonparametric regression. *Annals of Statistics*, 10(4):1040–1053.

Tarjan, R. (1983). *Data Structures and Network Algorithms*. Society for Industrial and Applied Mathematics.

Tenenbaum, J., de Silva, V., and Langford, J. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323.

Titterington, D., Smith, A., and Makov, U. (1985). *Statistical Analysis of Finite Mixture Distributions*. John Wiley.

Urner, R., Wulff, S., and Ben-David, S. (2013). PLAL: Cluster-based active learning. In *Proceedings of the 26th Annual Conference on Learning Theory*.

Vattani, A. (2009a). The hardness of k-means clustering in the plane. *Manuscript*.

Vattani, A. (2009b). k-means requires exponentially many iterations even in the plane. In *Proceedings of Symposium on Computational Geometry*.

Vempala, S. and Wang, G. (2004). A spectral algorithm for learning mixtures of distributions. *Journal of Computer and Systems Sciences*, 68(4):841–860.

Wishart, D. (1969). Mode analysis: a generalization of nearest neighbor which reduces chaining effects. In *Proceedings of the Colloquium on Numerical Taxonomy held in the University of St. Andrews*, pages 282–308.

Wong, M. and Lane, T. (1983). A kth nearest neighbour clustering procedure. *Journal of the Royal Statistical Society Series B*, 45(3):362–368.

Wu, C. (1983). On the convergence properties of the EM algorithm. *Annals of Statistics*, 11:95–103.

Zhu, X., Lafferty, J., and Ghahramani, Z. (2003). Combining active learning and semi-supervised learning using gaussian fields and harmonic functions. In *ICML Workshop on the Continuum from Labeled to Unlabeled Data*.