Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)

Near-Optimal Active Learning of Halfspaces via Query Synthesis in the Noisy Setting

Lin Chen,[1,2] **Hamed Hassani,**[3] **Amin Karbasi**[1,2]
[1]Department of Electrical Engineering, [2]Yale Institute for Network Science, Yale University
[3]Computer Science Department, ETH Zürich
{lin.chen, amin.karbasi}@yale.edu, hamed@inf.ethz.ch

Abstract

In this paper, we consider the problem of actively learning a linear classifier through query synthesis where the learner can construct artificial queries in order to estimate the true decision boundaries. This problem has recently gained a lot of interest in automated science and adversarial reverse engineering for which only heuristic algorithms are known. In such applications, queries can be constructed *de novo* to elicit information (e.g., automated science) or to evade detection with minimal cost (e.g., adversarial reverse engineering). We develop a general framework, called *dimension coupling* (DC), that 1) reduces a d-dimensional learning problem to $d-1$ low-dimensional sub-problems, 2) solves each sub-problem efficiently, 3) appropriately aggregates the results and outputs a linear classifier, and 4) provides a theoretical guarantee for all possible schemes of aggregation. The proposed method is proved resilient to noise. We show that the DC framework avoids the curse of dimensionality: its computational complexity scales linearly with the dimension. Moreover, we show that the query complexity of DC is near optimal (within a constant factor of the optimum algorithm). To further support our theoretical analysis, we compare the performance of DC with the existing work. We observe that DC consistently outperforms the prior arts in terms of query complexity while often running orders of magnitude faster.

1 Introduction

In contrast to the passive model of supervised learning, where all the labels are provided without any interactions with the learning mechanism, the key insight in active learning is that the learning algorithm can perform significantly better if it is allowed to choose which data points to label. This approach has found far-reaching applications, including the classical problems in AI (e.g., classification (Tong and Koller 2002), information retrieval (Tong and Chang 2001), speech recognition (Hakkani-Tur, Riccardi, and Gorin 2002)) as well as the modern ones (e.g., interactive recommender systems (Karbasi, Ioannidis, and Massoulie 2012), optimal decision making (Javdani et al. 2014)), and optimal information gathering (Chen et al. 2015). In all the above applications, the unlabeled data are usually abundant and easy to obtain, but training labels are either time-consuming or expensive to acquire (as they require asking an expert).

Throughout this paper, our objective is to actively learn an unknown halfspace $H^* = \{x \in \mathbb{R}^d : \langle h^*, x\rangle > 0\}$ via *query synthesis* (a.k.a. membership queries), where $\langle\cdot,\cdot\rangle$ denotes the standard inner product of the Euclidean space and h^* is the unit normal vector of the halfspace we want to learn. We would like to note that learning the halfspace H^* is mathematically equivalent to learning its unit normal vector h^*; therefore we focus on learning h^* hereinafter. In addition, it should be noted that using the kernel trick we can easily extend the halfspace learning to more complex (e.g., nonlinear) decision boundaries (Shawe-Taylor and Cristianini 2004).

The hypothesis space $\mathcal{H}$, which consists of all possibilities of unit normal vectors, is the unit sphere $S^{d-1} = \{x \in \mathbb{R}^d : \|x\| = 1\}$, where $\|\cdot\|$ denotes the standard Euclidean norm.

In active learning of halfspaces via query synthesis, the algorithm is allowed to query whether any point x in $\mathbb{R}^d$ resides in the true halfspace. When the algorithm queries x, the true outcome is $\mathbf{sign}(\langle h^*, x\rangle) \in \{1, -1\}$. When $\mathbf{sign}(\langle h^*, x\rangle) = 1$, it means that $x \in H^*$; otherwise, $x \notin H^*$. We should note here that the only information we obtain from a query is the *sign* of the inner product rather than the value. For example, the queries of the form $\mathbf{sign}(\langle h^*, e_i\rangle)$, where e_i is the ith standard basis vector, will only reveal the *sign* of the ith component of h^* (and nothing further about its value).

In the noiseless setting, we observe the true outcome of the query, i.e. $\mathbf{sign}\langle h^*, x\rangle \in \{1, -1\}$. In the noisy setting, the outcome is a flipped version of the true sign with independent flip probability ρ. That is, denoting the outcome by y we have $y \in \{-1, 1\}$ and $\Pr[y \neq \mathbf{sign}\langle h^*, x\rangle] \triangleq \rho < 1/2$.

Since the length of the selected vector x will not affect the outcome of the query, we only query the points on the unit sphere $S^{d-1} = \{x \in \mathbb{R}^d : \|x\| = 1\}$. Hence, we term $\mathcal{X} = S^{d-1}$ as the *query space*.

Given $\epsilon, \delta > 0$, we would like to seek an active learning algorithm that (i) adaptively selects vectors $x_1, x_2, \ldots \in \mathcal{X}$, (ii) observes the (noisy) responses to each query $\mathbf{sign}\langle h^*, x_i\rangle$, (iii) and outputs, using as few queries as possible, an estimate $\hat{h}$ of h^* such that $\|\hat{h} - h^*\| < \epsilon$ with probability at least $1 - \delta$.

Our main contribution in this paper is to develop a noise

resilient active learning algorithm that has access to *noisy* membership queries. To the best of our knowledge, we are the first to show a near-optimal algorithm that outperforms in theory and practice the naive repetition mechanism and the recent spectral heuristic methods (Alabdulmohsin, Gao, and Zhang 2015). Specifically, we develop a framework, called Dimension Coupling (DC), with the following guarantees. Its query complexity is $\tilde{O}(d(\log\frac{1}{\epsilon}+\log\frac{1}{\delta}))$[1] and its computational complexity is $\tilde{O}(d(\log\frac{1}{\epsilon}+\log\frac{1}{\delta})^2)$. In particular, in the noiseless setting ($\rho = 0$), both its computational complexity and query complexity are $\tilde{O}(d\log\frac{1}{\epsilon})$. Note that in both settings the computational complexity scales linearly with the dimension. Moreover, the query complexity in both settings is near-optimal. Our empirical experiments demonstrate that DC runs orders of magnitude faster than the existing methods.

The rest of the paper is structured as follows. In Section 2, we start with investigating this problem in the 2-dimensional case and present an algorithm called DC^2. Then in Section 3 we generalize it to the d-dimensional case and present a general framework called DC. Empirical results are shown in Section 4. We extensively review related literature in Section 5. *All proofs are provided in the extended version (Chen, Hassani, and Karbasi 2016).*

2 DC^2: Solving the 2-Dimensional Problem

To gain more intuition before studying the general d-dimensional problem, it might be beneficial to study a special case where the dimension is two. In other words, we study in this section how to learn the normalized projection of the true unit normal vector $h^* \in \mathbb{R}^d$ onto $\text{span}\{e_1, e_2\}$, where $e_1, e_2 \in \mathbb{R}^d$ are two orthonormal vectors and $\text{span}\{e_1, e_2\}$ is the linear subspace spanned by e_1 and e_2. We should note here that the underlying space is still d-dimensional (i.e., $\mathbb{R}^d$) but our goal is not to learn h^* *per se* but its normalized projection onto a 2-dimensional subspace.

Formally, given two orthonormal vectors e_1, e_2 we denote the (normalized) projection of h^* onto $\text{span}\{e_1, e_2\}$ by $h^\perp$, i.e.,

$$h^\perp = \frac{\langle h^*, e_1\rangle e_1 + \langle h^*, e_2\rangle e_2}{\|\langle h^*, e_1\rangle e_1 + \langle h^*, e_2\rangle e_2\|_2}. \quad (1)$$

Our objective is to find a unit vector $\hat{e} \in \text{span}\{e_1, e_2\}$ such that $\|\hat{e} - h^\perp\| < \epsilon$. In fact, we require the latter to hold with probability at least $1 - \delta$.

We should emphasize that noise, characterized by independent flip probability ρ, is generally present. In the 2-dimensional problem, one may propose to use the simple binary search (a detailed discussion with examples is presented in the extended version) to find a unit vector $\hat{e}$ that resides ϵ-close to $h^\perp$. To make it noise-tolerant, when the binary search algorithm queries a point, say x_i, we query it R times to obtain R noisy versions of $\textbf{sign}\langle h^*, x_i\rangle$ and view the majority vote of the noisy versions as the true outcome (Kääriäinen 2006; Karp and Kleinberg 2007; Nowak 2011). We call this method *repetitive querying*. However, its query complexity is $O(\log(1/\epsilon)(\log\log(1/\epsilon) + \log(1/\delta))$, which is suboptimal both theoretically (proof in the extended version) and empirically (referred to as REPETITIVE-DC in Section 4).

[1] We use the $\tilde{O}$ notation to ignore logarithmic factors. In Section 5, the $\tilde{O}$ notation also ignores terms dependent on δ.

Algorithm 1 DC^2

Input: orthonormal vectors e_1, e_2, estimation error at most ϵ, success probability at least $1 - \delta$.
Output: a unit vector $\hat{e}$ which is an estimate for the normalized orthogonal projection of h^* onto $\text{span}\{e_1, e_2\}$.

1: Set $p_0(h)$ to be uniform, i.e., $\forall h \in S^1 : p_0(h) = 1/2\pi$.
2: **for** $m = 1$ **to** $T_{\epsilon,\delta}$ **do**
3: Find a vector $x_m \in S^1$ which is a solution to the following equation: $\int_{S^1} \textbf{sign}\langle x, h\rangle\, p_{m-1}(h)dh = 0$. If there are multiple solutions, choose one arbitrarily.
4: Ask from the oracle the value of $\textbf{sign}\langle x_m, h^*\rangle$.
5: Based on the (noisy) response obtained from the oracle, update the distribution $p_{m-1}(h)$ to $p_m(h)$.
6: **end for**
7: **return** $\hat{e} = \arg\max_{h\in S^1} p_{T_{\epsilon,\delta}}(h)$.

As a result, instead, we will present a Bayesian algorithm termed DC^2 that solves this 2-dimensional problem with query complexity $O(\log(1/\epsilon) + \log(1/\delta))$. Recall that any unit vector inside $\text{span}\{e_1, e_2\}$, e.g., $h^\perp$, can equivalently be represented as a pair (c_1, c_2) on the two-dimensional unit circle S^1 (e.g., $h^\perp = c_1e_1 + c_2e_2$ and $c_1^2 + c_2^2 = 1$). To simplify notation, we use a point $(c_1, c_2) \in S^1$ and its corresponding unit vector $c_1e_1 + c_2e_2$ interchangeably. In this setting, it is easy to see that for any $x \in \text{span}\{e_1, e_2\}$

$$\textbf{sign}\langle x, h^*\rangle = \textbf{sign}\langle x, h^\perp\rangle. \quad (2)$$

We take a Bayesian approach. In the beginning, when no queries have been performed, DC^2 assumes no prior information about the vector $h^\perp$. Therefore, it takes the uniform distribution on S^1 (with pdf $p_0(h) = \frac{1}{2\pi}$) as its prior belief about $h^\perp$. After performing each query, the posterior (belief) about $h^\perp$ will be updated according to the observation. We let $p_m(h)$ denote the (pdf of the) posterior after performing the first m queries. In this manner, DC^2 runs in total of $T_{\epsilon,\delta}$ rounds, where in each round a specific query is selected and posed to the oracle. The number $T_{\epsilon,\delta}$ will be specified later (see Theorem 1). Upon the completion of round $T_{\epsilon,\delta}$, the algorithm returns as its final output a vector $\hat{e} \in S^1$ that maximises the posterior pdf $p_{T_{\epsilon,\delta}}(h)$. If there are multiple such maximisers, it picks one arbitrarily. We now proceed with a detailed description of DC^2 (a formal description is provided in Algorithm 1).

As shown in Algorithm 1, at each round, say round $m+1$, the algorithm maintains and updates the distribution p_m that encodes its current belief in the true location of $h^\perp$. We should note here that these distributions can be stored efficiently and as a result the vector x_{m+1} can be computed efficiently. Indeed, (the pdf of) p_m is piecewise constant on the unit circle (see Figure 1). More precisely, at any round m, there are at most $2m$ points $u_1, u_2, \cdots, u_{2m}$ that are ordered clock-wise on the unit-circle and p_m is constant when restricted to each of the sectors $[u_i, u_{i+1})$. The piecewise

constant property of the pdf of p_m can be established by induction on m. Recall that the initial distribution p_0 is uniform and thus piecewise constant. The Bayesian update step (line 5 of Algorithm 1) preserves this property when the algorithm updates the distribution $p_m(h)$ to $p_{m+1}(h)$. We will show why this is true when we discuss the Bayesian update step in detail.

At round $m+1$, in order to find x_{m+1} (see line 3 of Algorithm 1), DC^2 first finds a line that passes through the centre of S^1 and cuts S^1 into two "halves" which have the same measure with respect to p_m. Note that finding such a line can be done in $O(m)$ steps because p_m has the piecewise constant property. Once such a line is found, it is then easy to see that x_{m+1} can be any of the two points orthogonal to the line. As a result, DC^2 at round $m+1$ can find x_{m+1} in $O(m)$ operations. We denote the half-circle containing x_{m+1} by R^+ and the other half by R^-. We refer to Figure 1 for a schematic illustration.

The key step in Algorithm 1 is the Bayesian update (line 5). Once a noisy response to the query $\textbf{sign}\langle x_{m+1}, h^*\rangle$ is obtained (line 4)), the probability distribution p_m will be updated to p_{m+1} in the following way. First, consider the event that the outcome of $\textbf{sign}\langle x_{m+1}, h^*\rangle$ is $+1$. We have $p_m(\textbf{sign}\langle x_{m+1}, h^*\rangle = +1) = (1-\rho)\ p_m(R^+) + \rho\ p_m(R^-) = 1/2$, and similarly $p_m(\textbf{sign}\langle x_{m+1}, h^*\rangle = -1) = 1/2$. Therefore, by Bayes theorem we obtain the following update rules for p_{m+1}. If we observe that the outcome of $\textbf{sign}\langle x_{m+1}, h^*\rangle$ is $+1$, then for $h \in R^+$ we have $p_{m+1}(h) = 2(1-\rho)p_m(h)$ and for $h \in R^-$ we have $p_{m+1}(h) = (2\rho)p_m(h)$. Also, if we observe that the outcome of $\textbf{sign}\langle x_{m+1}, h^*\rangle$ is -1, then for $h \in R^+$ we have $p_{m+1}(h) = (2\rho)p_m(h)$ and for $h \in R^-$ we have $p_{m+1}(h) = 2(1-\rho)p_m(h)$. Note that the factor of 2 here is due to the normalization. It is easy to verify that p_{m+1} is also a piecewise constant distribution (now on $2(m+1)$ sectors; see Figure 1).

Theorem 1 shows that after $T_{\epsilon,\delta} = O(\log\frac{1}{\epsilon} + \log\frac{1}{\delta})$ rounds, with probability at least $1-\delta$, DC^2 outputs a unit vector $\hat{e} \in \text{span}\{e_1, e_2\}$ such that $\|\hat{e} - h^\perp\| < \epsilon$. Also, as discussed above, the computational complexity of DC^2 is $O(T^2_{\epsilon,\delta})$, i.e., $O((\log\frac{1}{\epsilon} + \log\frac{1}{\delta})^2)$.

Theorem 1. *When the independent flip probability is ρ, having*

$$T_{\epsilon,\delta} \geq M + \max\{T_0, T_1, T_2, T_3\} = O(\log\frac{1}{\epsilon} + \log\frac{1}{\delta}) \quad (3)$$

is sufficient to guarantee that DC^2 *outputs with probability at least $1-\delta$ a vector that is within a distance ϵ of $h^\perp$. Here, we have $M = \lceil \frac{2\log\frac{2}{\delta}}{-\log(4\rho(1-\rho))} \rceil$, $T_0 = \frac{8\log\frac{2}{\delta}}{\log(2(1-\rho))}$, $T_1 = \frac{8\log\frac{1}{8\pi\epsilon}}{\log(2(1-\rho))}$, $T_2 = \frac{8}{\log(2(1-\rho))}\left(\log(2M) + \log(\frac{4}{\log(2(1-\rho))})\right)$ and $T_3 = \frac{24\rho\log^2\frac{1-\rho}{\rho}}{\log^2(2(1-\rho))}\left(\log(M) + \log(\frac{4}{\delta})\right)$.*

We would like to remark that when the independent flip probability ρ is 0 (i.e., in the noiseless case), the algorithm DC^2 reduces to the binary search. If we let $T_{\epsilon,\delta} = \lceil \log_2\frac{\pi}{\epsilon} \rceil$, then DC^2 outputs a vector that is within a distance ϵ of $h^\perp$. We present a detailed discussion with examples in the extended version.

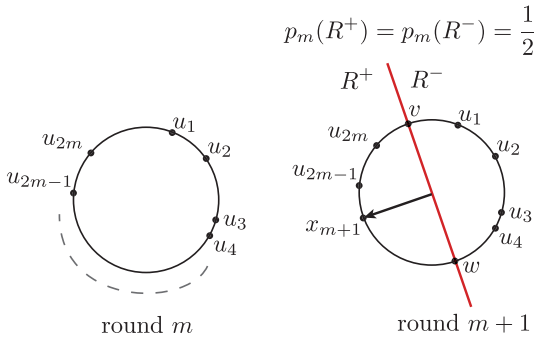


Figure 1: Upon the completion of round m (left figure), the distribution (pdf of) p_m is constant over each of the sectors $[u_i, u_{i+1})$. In the next round (right figure), in order to find x_{m+1}, DC^2 first finds a diagonal line (red line) which separates two half-circles (R^+ and R^-) that each has measure $1/2$ w.r.t p_m. The vector x_{m+1} will then be one of the two points on the unit circle that are orthogonal to this line. For updating p_m to p_{m+1}, we note that all the points inside R^+ get the same factor (either 2ρ or $2(1-\rho)$ depending on the outcome of the query). The same is true for R^-. Thus, p_{m+1} is again a piecewise constant pdf but now on $2(m+1)$ sectors.

A few comments are in order: The above guarantee for DC^2 holds with probability one and thus the parameter δ is irrelevant in the noiseless setting. Furthermore, during each round of DC^2, the distribution p_m can be represented by only two numbers (the starting and ending points of the version space after m rounds R_m, which is a sector), and the vector x_m can be computed efficiently (it is the orthogonal vector to the midpoint of R_m). Therefore, assuming one unit of complexity for performing the queries, DC^2 can be implemented with complexity $O(T_{\epsilon,\delta})$, i.e., $O(\log(1/\epsilon))$.

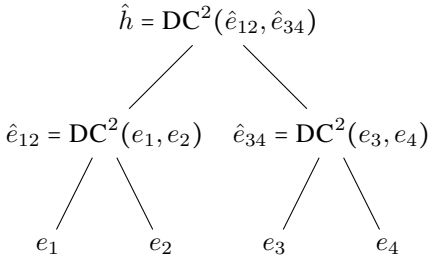
3 Dimension Coupling Based Framework

In Section 2, we devise an algorithm, called DC^2 $(e_1, e_2, \epsilon, \delta)$, that takes as input two orthonormal vectors e_1, e_2, uses noisy responses to queries of the form $\textbf{sign}\langle x, h^*\rangle$, and outputs with probability at least $1-\delta$ a vector $\hat{e}$ with the following three properties:

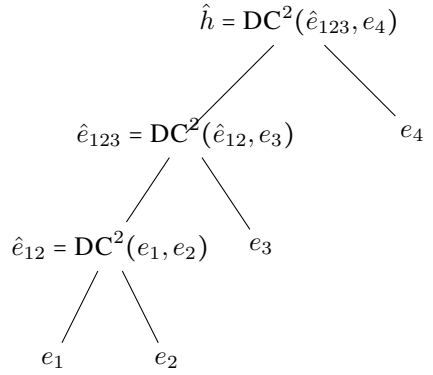
$$\hat{e} \in \text{span}\{e_1, e_2\}, \|\hat{e}\| = 1, \|\hat{e} - \frac{\langle h^*, e_1\rangle e_1 + \langle h^*, e_2\rangle e_2}{\|\langle h^*, e_1\rangle e_1 + \langle h^*, e_2\rangle e_2\|}\| < \epsilon.$$

In other words, the unit vector $\hat{e}$ is within a distance ϵ to the (normalized) projection of h^* onto the subspace $\text{span}\{e_1, e_2\}$. In the current section, we explain a framework DC that estimates h^* using at most $d-1$ calls to DC^2 (a formal description will be given in Algorithm 2 later).

Let us begin our discussion with a motivating example. Let $\{e_1, e_2, \ldots, e_d\}$ be an orthonormal basis of $\mathbb{R}^d$. Suppose that h^* has the form $h^* = \sum_{i=1}^d c_i e_i$, where $\{e_i\}_{i=1}^d$ is an arbitrarily chosen orthonormal basis for $\mathbb{R}^d$. We assume w.l.o.g. that h^* is normalized (i.e., $\sum_{i=1}^d c_i^2 = 1$). Our objective is then to learn the coefficients $\{c_i\}_{i=1}^d$ within a given precision by using the noisy responses to the selected sign queries. The key insight here is that this task can be partitioned in



(a) Scheme 1: a balanced full binary tree



(b) Scheme 2: an unbalanced full binary tree

Figure 2: Two possible divide-and-conquer schemes for a 4-dimensional problem. Each scheme can be represented by a full binary tree of 4 leaf nodes.

a divide-and-conquer fashion into many smaller tasks, each involving a few dimensions. The final answer (the values of $\{c_i\}_{i=1}^d$) will then be obtained by aggregating the answers of these subproblems.

In Figure 2, we present two possibilities of divide-and-conquer schemes for a 4-dimensional problem. In fact, each scheme corresponds to a full binary tree of 4 leaf nodes.

Assume $h^* = c_1e_1 + c_2e_2 + c_3e_3 + c_4e_4$, where e_i's are the standard basis vectors for $\mathbb{R}^4$. Define $e_{12} = \frac{c_1e_1+c_2e_2}{\sqrt{c_1^2+c_2^2}}$, $e_{34} = \frac{c_3e_3+c_4e_4}{\sqrt{c_3^2+c_4^2}}$. Note here that e_{12} is the (normalized) orthogonal projection of h^* onto $\mathrm{span}\{e_1, e_2\}$ and e_{34} is the (normalized) orthogonal projection of h^* onto $\mathrm{span}\{e_3, e_4\}$. Consider the following procedure to learn h^*: first find out what e_{12} and e_{34} are, and then use the relation $h^* = \sqrt{c_1^2 + c_2^2}e_{12} + \sqrt{c_3^2 + c_4^2}e_{34}$ to find h^* based on the orthonormal vectors e_{12}, e_{34}. By this procedure, the original "four-dimensional" problem has been broken into three "two-dimensional" problems.

This procedure is illustrated in Figure 2a. We first call $\mathrm{DC}^2(e_1, e_2)$ to obtain an estimate $\hat{e}_{12}$ for e_{12}; then we call $\mathrm{DC}^2(e_3, e_4)$ to obtain an estimate $\hat{e}_{34}$ for e_{34}; finally we call $\mathrm{DC}^2(\hat{e}_{12}, \hat{e}_{34})$ to obtain an estimate $\hat{h}$ for h^*. Another *unbalanced* scheme is illustrated in Figure 2b.

For general d, the idea is similar: We break the problem into at most $d-1$ "two-dimensional" problems that each can be solved efficiently. Again, each divide-and-conquer scheme corresponds to a full binary tree of d leaf nodes.

Consider the decomposition $h^* = \sum_{i=1}^d c_ie_i$. Without loss of generality, suppose that the first two leaf nodes to be combined are e_1 and e_2. We can write

$$h^* = \sum_{i=1}^d c_ie_i = \hat{c}_{12}\frac{c_1e_1 + c_2e_2}{\sqrt{c_1^2 + c_2^2}} + \sum_{i=3}^d c_ie_i, \tag{4}$$

where in the last step we have taken $\hat{c}_{12} \triangleq \sqrt{c_1^2 + c_2^2}$. Now, note that $\hat{e}_{12} \triangleq \frac{c_1e_1+c_2e_2}{\sqrt{c_1^2+c_2^2}}$ is the normalized orthogonal projection of h^* onto $\mathrm{span}\{e_1, e_2\}$. Hence, by using DC^2 $(e_1, e_2, \epsilon, \delta)$ we can obtain, with probability at least $1-\delta$, a good approximation $\hat{e}_{12}$ (within a distance ϵ) of this projection. Therefore, for small enough ϵ we have $h^* \approx \hat{c}_{12}\hat{e}_{12} + \sum_{i=3}^d c_ie_i$. Since h^* is now expressed (approximately) in terms of $d-1$ orthonormal vectors $\{\hat{e}_{12}, e_3, e_4, \ldots, e_d\}$, we have effectively reduced the dimensionality of problem from d to $d-1$. The idea is then to repeat the same procedure as in (4) to the newly obtained representation of h^*. Hence, by repeating this procedure $d-1$ times in total we will reach a vector which is the final approximation of h^*.

We present this general method in Algorithm 2.

Algorithm 2 Dimension Coupling (DC)

Input: an orthonormal basis $E = \{e_1, e_2, \ldots, e_d\}$ of $\mathbb{R}^d$.
Output: a unit vector $\hat{h}$ which is an estimate for h^*.

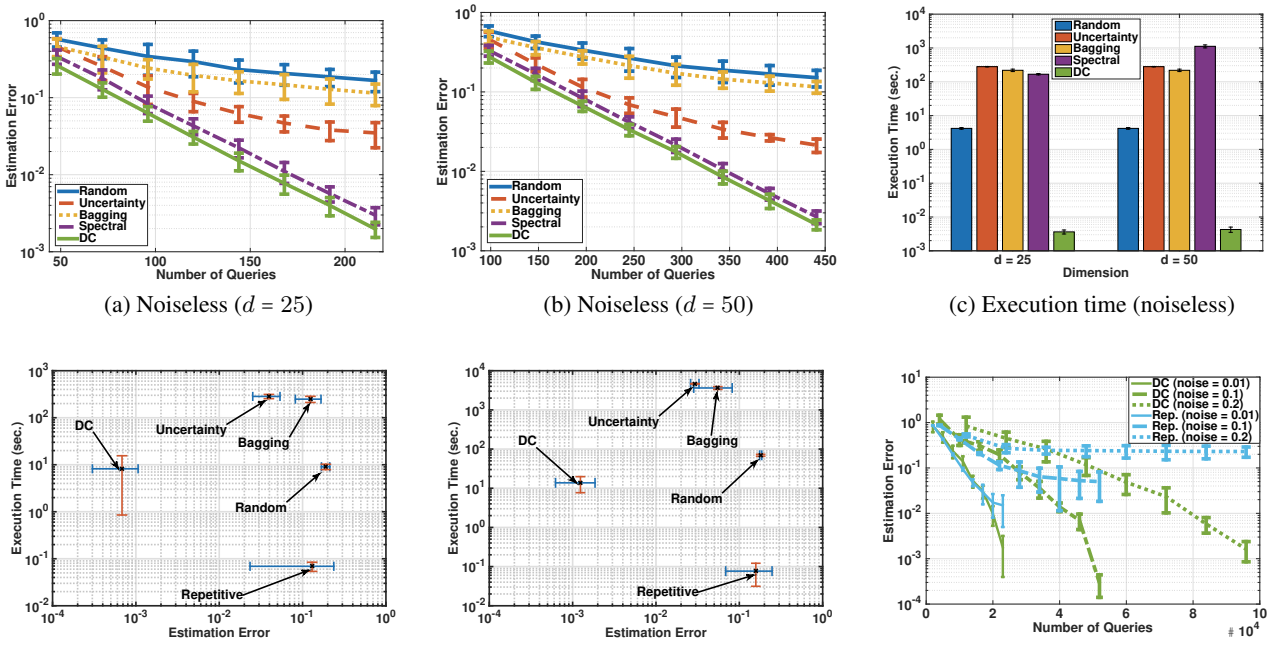
1: **for** $j \leftarrow 1$ **to** $d-1$ **do**
2: Replace any two vectors e' and e'' in E with the vector DC^2 $(e', e'', \epsilon, \delta)$.
3: **end for**
4: Let $\hat{h}$ be the only remaining vector in E.
5: **return** $\hat{h}$

Theorem 2. *For* DC *(outlined in Algorithm 2) and any of its divide-and-conquer scheme represented by a full binary tree, we have:*

1. DC *will call the two-dimensional subroutine* DC^2 $d-1$ *times.*
2. *Provided that the output of* DC^2 *is with probability* $1-\delta$ *within distance* ϵ *of the true value and* $\epsilon \le 5/18$*,* DC *ensures an estimation error of at most* $5\epsilon(d-1)$ *with probability at least* $1-\delta(d-1)$.

As a result of Theorem 2, if we desire the framework DC to estimate h^* within distance $\tilde{\epsilon}$ and with probability at least $1-\tilde{\delta}$, then it is enough to fix the corresponding parameters of DC^2 to $\epsilon = \frac{\tilde{\epsilon}}{5(d-1)}$ and $\delta = \frac{\tilde{\delta}}{d-1}$.

Theorem 2 indicates that DC requires $\tilde{O}(d(\log\frac{1}{\epsilon} + \log\frac{1}{\delta}))$ queries, since each call to DC^2 needs $O(\log\frac{1}{\epsilon} + \log\frac{1}{\delta})$ queries. Recall that the computational complexity of DC^2 is $O((\log\frac{1}{\epsilon} + \log\frac{1}{\delta})^2)$. Hence, DC has computational complexity $\tilde{O}(d(\log\frac{1}{\epsilon} + \log\frac{1}{\delta})^2)$. As a special case, if in absence of noise, both the query complexity and time complexity of DC are $\tilde{O}(d\log\frac{1}{\epsilon})$.



(a) Noiseless ($d = 25$) (b) Noiseless ($d = 50$) (c) Execution time (noiseless)

(d) Noisy ($d = 25$) (e) Noisy ($d = 50$) (f) Noisy ($d = 1000$)

Figure 3: Figures 3a and 3b show the estimation error in the noiseless setting as we increase the number of queries, for $d = 25$ and 100, respectively. Figure 3c shows the corresponding execution times. Figure 3d and 3e show the scatter plots of the execution time and the estimation error of different methods for $d = 25, 50$ and the noise level $\rho = 0.1$. We allow each algorithm to use a budget of 800 and 1800 queries in Figure 3d and 3e, respectively. Figure 3f presents the estimation error of DC and REPETITIVE-DC as we increase the number of queries for $d = 1000$ and noise levels $\rho = 0.01, 0.1, 0.2$.

4 Empirical Results

In this section, we extensively evaluate the performance of DC against the following baselines:

RANDOM-SAMPLING: Queries are generated by sampling uniformly at random from the unit sphere S^{d-1}.

UNCERTAINTY-SAMPLING: Queries are sampled uniformly at random from the orthogonal complement of w, where w is the vector learned by linear SVM.

QUERY-BY-BAGGING: The bag size is set to 20 and 1000 queries are generated at each iteration. The query with the largest disagreement is picked (Abe and Mamitsuka 1998).

SPECTRAL: The version space is approximated by the largest ellipsoid consistent with all previous query-label pairs. Then, at each iteration a query is selected to approximately halve the ellipsoid (Alabdulmohsin, Gao, and Zhang 2015).

REPETITIVE-DC: In the noisy setting, one easy way to apply DC is to query each point R times and use the majority rule to determine its label; i.e., the combination of repetitive querying (Section 2) and the DC framework (Section 3).

Our metrics to compare different algorithms are: a) estimation error, b) query complexity, and c) execution time. In particular, as we increase the number of queries we measure the average estimation errors and execution times for all the baselines (with 90% confidence intervals). By nature, in active learning via query synthesis, all data points and queries are generated synthetically. For all the baselines, we used the fastest available implementations in MATLAB.

Noiseless setting: Figures 3a and 3b (with dimension $d = 25$ and 50, respectively) show that in terms of estimation error, DC outperforms all other baselines, and significantly outperforms RANDOM-SAMPLING, UNCERTAINTY-SAMPLING and QUERY-BY-BAGGING. Note that the estimation errors are plotted in log-scales. In terms of execution times, we see in Fig. 3c that DC runs three orders of magnitude faster than other baselines. Training an SVM at each iteration for RANDOM-SAMPLING, UNCERTAINTY-SAMPLING and QUERY-BY-BAGGING comes with a huge computational cost. Similarly, SPECTRAL requires solving a convex optimization problem at each iteration; thus its performance drastically deteriorates as the dimension increases, which makes it infeasible for many practical problems.

Noisy setting: We set the noise level to $\rho = 0.1$ and compare the performance of DC against RANDOM-SAMPLING, UNCERTAINTY-SAMPLING, QUERY-BY-BAGGING, and REPETITIVE-DC. As mentioned in (Alabdulmohsin, Gao, and Zhang 2015), and we have also observed in our experiments, SPECTRAL does not work even for small amounts of noise as it incorrectly shrinks the version space and misses the true linear separator; therefore it is excluded here. We see again in Figures 3d and 3e (for $d = 25$ and 50) that DC significantly outperforms all other methods in terms of estimation error. More precisely, using the same number of

queries, the estimation error of DC is around two orders of magnitude smaller than other baselines. We can also observe from these two figures that DC still runs around 100 times faster than RANDOM-SAMPLING, UNCERTAINTY-SAMPLING, and QUERY-BY-BAGGING. Clearly, DC has a higher computational cost than REPETITIVE-DC, as DC performs a Bayesian update after each query. Finally, as we increase the dimension to $d = 1000$, RANDOM-SAMPLING, UNCERTAINTY-SAMPLING, and QUERY-BY-BAGGING become significantly slower. Hence, in Figure 3f we only show how the estimation error (for noise levels $\rho = 0.01, 0.1, 0.2$) decreases for DC and REPETITIVE-DC with more queries. It can be observed from Figure 3f that consuming the same number of queries, DC can achieve an estimation error from one order (when the noise intensity is very small) to three orders of magnitude (when the noise intensity is 0.2) smaller than that of REPETITIVE-DC.

5 Related Work

The sample complexity of learning a hypothesis was traditionally studied in the context of probably approximately correct (PAC) learning (Valiant 1984). In PAC learning theory, one assumes that a set of hypotheses $\mathcal{H}$ along with a set of unlabeled data points $\mathcal{X}$ are given, where each data point $x \in \mathcal{X}$ is drawn i.i.d. from some distribution D. Classical PAC bounds then yield the sample complexity (i.e., the number of required i.i.d. examples) from D to output a hypothesis $h \in \mathcal{H}$ that will have *estimation error* at most ϵ with probability at least $1 - \delta$, for some fixed $\epsilon, \delta > 0$. Here, the estimation error is defined as $\epsilon = \Pr_{x \sim D}[h(x) \neq h^*(x)]$, where h^* is the unknown true hypothesis. In the *realizable* case of learning a halfspace, i.e., when $h^* \in \mathbb{R}^d$ perfectly separates the data points into positive and negative labels, it is known that with $\tilde{O}(d/\epsilon)$ i.i.d. samples one can find a linear separator with an estimation error ϵ. The main advantage of using active learning methods, i.e., sequentially querying data points, is to reduce the sample complexity exponential fast, ideally to $\tilde{O}(d \log(1/\epsilon))$. In fact, a simple counting argument based on sphere packing shows that any algorithm needs $\Omega(d \log(1/\epsilon))$ examples to achieve an estimation error of ϵ (Dasgupta, Kalai, and Monteleoni 2009).

For $d = 2$ and when the distribution is uniform over the unit sphere S^1 it is very easy to see that the halving or bisection leads to $\tilde{O}(\log(1/\epsilon))$. By using the same halving method, one can in principle extend the result to any dimension d. To do so, we need to carefully construct the version space (i.e., the set of hypotheses consistent with the queries and outcomes) at each iteration and then find a query that halves the volume (in the uniform case) or the density (in the general case if the distribution is known) (Dasgupta 2004). Finding such a query in high dimension is very challenging.

One very successful approach that does not suffer from the aforementioned computational challenge is pool-based active learning (Settles 2010), where instead of ideally halving the space, effective approaximations are performed. Notable algorithms are *uncertainty sampling* (Lewis and Gale 1994) and query-by-committee (QBC) (Freund et al. 1997). In fact, our problem is closely related to learning homogeneous linear separators under the uniform distribution in the pool-based setting. This problem is very well understood and there exist efficient pool-based algorithms (Balcan, Broder, and Zhang 2007; Dasgupta, Kalai, and Monteleoni 2005; Dasgupta and Hsu 2008). In particular, Dasgupta et al. (Dasgupta, Kalai, and Monteleoni 2009) presented an efficient perceptron-based algorithm that achieve a near-optimal query complexity. Similar results can be obtained under log-concave distributions (Balcan and Long 2013). Most of the pool-based methods require to have access to $\tilde{O}(1/\epsilon)$ number of unlabeled samples in each iteration or otherwise they perform very poorly (Balcan, Broder, and Zhang 2007; Dasgupta, Kalai, and Monteleoni 2009). This means that in order to have exponential guarantee in terms of sample complexity, we need to grow the pool size exponentially fast (note that there is no need to store all of these points). Moreover, with a few exceptions (Awasthi, Balcan, and Long 2014; Balcan, Beygelzimer, and Langford 2006) pool-based learning of linear separators in the noisy setting has been much less studied and the dependency of sample complexity on noise is not very well understood.

An attractive alternative to the pool-based framework is query synthesis where we have access to membership queries (Angluin 1988)): a learner can request for any unlabeled data instance from the input space, including queries that the learner synthesizes from scratch. This way the pool size limitation is entirely eliminated. In many recent applications, ranging from automated science (King et al. 2009), to robotics (Cohn, Ghahramani, and Jordan 1996), and to adversarial reverse engineering (Lowd and Meek 2005), query synthesis is the appropriate model. For instance, in security-sensitive applications (e.g., spam filters and intrusion detection systems) that routinely use machine learning tools, a growing concern is the ability of adversarial attacks to identify the blind spots of the learning algorithms. Concretely, classifiers are commonly deployed to detect miscreant activities. However, they are attacked by adversaries who generate exploratory queries to elicit information that in return allows them to evade detection (Nelson et al. 2012). In this work, we show how an adversary can use active learning methods by making synthetically *de novo* queries and thus identify the linear separator used for classification. We should emphasize that in active learning via *synthesized* queries the learning algorithm can query the label of *any points* in order to explore the hypothesis space. In the noiseless setting (if we ignore the dependency of the pool size on $\tilde{O}(\log(1/\epsilon))$), one can potentially use the pool-based algorithms (under the uniform distribution). Our main contribution in this paper is to develop a noise resilient active learning algorithm that has access to *noisy* membership queries. To the best of our knowledge, we are the first to show a near optimal algorithm that outperforms in theory and practice the naive repetition mechanism and the recent spectral heuristic methods (Alabdulmohsin, Gao, and Zhang 2015).

Acknowledgements

LC and AK were supported by DARPA Young Faculty Award (D16AP00046). HH was supported partially by ERC

StG 307036. LC thanks Zheng Wei for his support.

References

Abe, N., and Mamitsuka, H. 1998. Query learning strategies using boosting and bagging. In *ICML*, volume 1.

Alabdulmohsin, I.; Gao, X.; and Zhang, X. 2015. Efficient active learning of halfspaces via query synthesis. In *AAAI*, 2483–2489.

Angluin, D. 1988. Queries and concept learning. *Machine learning* 2(4):319–342.

Awasthi, P.; Balcan, M. F.; and Long, P. M. 2014. The power of localization for efficiently learning linear separators with noise. In *STOC*, 449–458. ACM.

Balcan, M.-F., and Long, P. M. 2013. Active and passive learning of linear separators under log-concave distributions. In *COLT*, 288–316.

Balcan, M.-F.; Beygelzimer, A.; and Langford, J. 2006. Agnostic active learning. In *ICML*, 65–72. ACM.

Balcan, M.-F.; Broder, A.; and Zhang, T. 2007. Margin based active learning. In *Learning Theory*. Springer. 35–50.

Chen, Y.; Hassani, S. H.; Karbasi, A.; and Krause, A. 2015. Sequential information maximization: When is greedy near-optimal? In *COLT*, volume 40.

Chen, L.; Hassani, H.; and Karbasi, A. 2016. Near-optimal active learning of halfspaces via query synthesis in the noisy setting. *arXiv preprint arXiv:1603.03515*. https://arxiv.org/abs/1603.03515.

Cohn, D. A.; Ghahramani, Z.; and Jordan, M. I. 1996. Active learning with statistical models. *JAIR*.

Dasgupta, S., and Hsu, D. 2008. Hierarchical sampling for active learning. In *ICML*, 208–215. ACM.

Dasgupta, S.; Kalai, A. T.; and Monteleoni, C. 2005. Analysis of perceptron-based active learning. In *COLT*, 249–263.

Dasgupta, S.; Kalai, A. T.; and Monteleoni, C. 2009. Analysis of perceptron-based active learning. *JMLR* 10(Feb):281–299.

Dasgupta, S. 2004. Analysis of a greedy active learning strategy. In *NIPS*, 337–344.

Freund, Y.; Seung, H. S.; Shamir, E.; and Tishby, N. 1997. Selective sampling using the query by committee algorithm. *Machine learning* 28(2-3):133–168.

Hakkani-Tur, D.; Riccardi, G.; and Gorin, A. 2002. Active learning for automatic speech recognition. In *ICASSP*, volume 4, IV–3904. IEEE.

Javdani, S.; Chen, Y.; Karbasi, A.; Krause, A.; Bagnell, J. A.; and Srinivasa, S. 2014. Near optimal bayesian active learning for decision making. In *AISTAT*, 430–438.

Kääriäinen, M. 2006. Active learning in the non-realizable case. In *ALT*, 63–77. Springer.

Karbasi, A.; Ioannidis, S.; and Massoulie, L. 2012. Comparison-based learning with rank nets. *ICML*.

Karp, R. M., and Kleinberg, R. 2007. Noisy binary search and its applications. In *SODA*, 881–890.

King, R. D.; Rowland, J.; Oliver, S. G.; Young, M.; Aubrey, W.; Byrne, E.; Liakata, M.; Markham, M.; Pir, P.; Soldatova, L. N.; et al. 2009. The automation of science. *Science* 324(5923):85–89.

Lewis, D. D., and Gale, W. A. 1994. A sequential algorithm for training text classifiers. In *SIGIR*, 3–12.

Lowd, D., and Meek, C. 2005. Adversarial learning. In *KDD*, 641–647.

Nelson, B.; Rubinstein, B. I.; Huang, L.; Joseph, A. D.; Lee, S. J.; Rao, S.; and Tygar, J. 2012. Query strategies for evading convex-inducing classifiers. *JMLR* 13(May):1293–1332.

Nowak, R. D. 2011. The geometry of generalized binary search. *IEEE Transactions on Information Theory* 57(12):7893–7906.

Settles, B. 2010. Active learning literature survey. *University of Wisconsin, Madison* 52(55-66):11.

Shawe-Taylor, J., and Cristianini, N. 2004. *Kernel methods for pattern analysis*. Cambridge university press.

Tong, S., and Chang, E. 2001. Support vector machine active learning for image retrieval. In *Proc. 9th ACM international conference on Multimedia*, 107–118.

Tong, S., and Koller, D. 2002. Support vector machine active learning with applications to text classification. *JMLR* 2:45–66.

Valiant, L. G. 1984. A theory of the learnable. *Communications of the ACM* 27(11):1134–1142.