Learning Using Local Membership Queries

Galit Bary

Submitted in partial fulfillment of the requirements
of the degree of Master of Science

Under the supervision of
Prof. Shai Shalev-Shwartz
Amit Daniely

September 2015

Rachel and Selim Benin
School of Computer Science and Engineering
The Hebrew University of Jerusalem
Israel

Abstract

Classic machine learning algorithms learn from labelled examples. For example, to design a machine translation system, a typical training set will consist of English sentences and their translation to French. There is a stronger model, in which the algorithm can also query for labels of new examples it creates. E.g, in the translation task, the algorithm can create a new English sentence, and request its translation from the user during training. This combination of examples and queries, that resembles human learning patterns, has been widely studied. Yet, despite many theoretical results, query algorithms are almost never used. One of the main causes for this is a report (Baum and Lang, 1992) on very disappointing empirical performance of a query algorithm. These poor results were mainly attributed to the fact that the algorithm queried for labels of examples that are artificial, and impossible to interpret by humans.

In this work we study a new model of *local* membership queries (Awasthi et al., 2012), which tries to resolve the problem of artificial queries. In this model, the algorithm is only allowed to query the labels of examples which are close to examples from the training set. E.g., in translation, the algorithm can change individual words in a sentence it has already seen, and then ask for the translation. In this model, the examples queried by the algorithm will be close to natural examples and hence, hopefully, will not appear as artificial or random. In this work we focus on 1-local membership queries (i.e., queries of distance 1 from an example in the training sample). We show that 1-local membership queries are already stronger than the standard learning model. We also present an experiment on a well known NLP task of sentiment analysis. In this experiment, the users were asked to provide, in a way that resembles 1-local queries, more information than merely indicating the label. We present results that illustrate that this extra information is beneficial in practice.

Acknowledgments

I would like to thank my advisor Prof. Shai Shalev-Shwartz for having me on his outstanding team and for his support and inspiration. I would also like to thank Amit Daniely, for his guidance and mentorship. His extensive knowledge and patience were invaluable. It has been a privilege to work with him.

To Alon Gonen, Nir Rosenfled, Yoav Wald, Yossi Arjevani, Nomi Vinokurov and Avishai Wagner for their remarkable friendship and counsel. To the NLP lab and especially Effi Levi for all his assistance in the empirical work and to my officemates Zahi Ajami and Dikla Cohn for their wonderful companionship.

I would like to thank my parents for all the love and support throughout the years, and to the Weisberg family for their help, especially Susan for her editorial comments. Last but not least, I would like to thank my husband Dov for his enduring support that is expressed on so many levels – encouraging me during difficult times, editing my drafts and providing home cooked meals.

Contents

1 Introduction

How do humans learn? Say we look at the process of a child learning how to recognize a cat. We can focus on two types of input. The first type of input is when a child's parent points at a cat and states "Look, a cat!". The second type of input is an answer to the child's frequent question "What is that?", which the child may pose when seeing a cat, but also when seeing a dog, a mouse, a rabbit, or any other small animal.

These two types of input were the basis for the learning model originally suggested in the celebrated paper "A theory of the learnable" (Valiant, 1984). In Valiant's learning model, the learning algorithm has access to two sources of information - EXAMPLES and ORACLE. The learning algorithm can call EXAMPLES to receive an example with its label (sampled from the "nature"). Additionally, the learning algorithm can use ORACLE, which provides the label of *any* example presented to it. With these two input types, we can look at two models of learning: learning using only calls for EXAMPLES, and learning using calls for both EXAMPLES and ORACLE. The first is the standard Probably Approximately Correct (PAC) model. The second is the so called PAC+MQ (Membership Queries) model. There has been a lot of theoretical work searching for the limits of the additional strength of membership queries. The use of membership queries in addition to examples was proven to be stronger than the standard PAC model in many cases (Angluin, 1987; Blum and Rudich, 1992; Bshouty, 1995; Jackson, 1994)(see section 2).

Despite that the MQ model seems much stronger, both intuitively and formally, it is rarely used in practice. This is commonly believed to result from the fact that in many cases it is not easy to implement MQ algorithms, that can create new and artificial examples to be labeled as part of the training phase. This problem of labeling artificial examples was highlighted by the experiment of Baum and Lang (1992). Baum and Lang implemented a membership query algorithm proposed by Baum (1991) for learning halfspaces . Their algorithm had very poor results, which was attributed to the fact that the algorithm created artificial and unnatural examples, which resulted in a noisy labeling. We elaborate on this experiment and criticize its conclusions in section 2.

A suggested solution to the problem of unnatural examples was proposed by Awasthi et al. (2012). They suggested a mid-way model of learning with queries, but only restricted ones. The queries that their model allows the algorithm to ask are only *local* queries, i.e., queries that are close in some sense to examples from the sample set. Hopefully, examples which are similar to natural examples will also appear to be natural, or at least close to natural, and in any case will be far from appearing random or artificial. In their work, Awasti et al. started to investigate the power and the limitations of this model of local queries. They proved positive results on learning sparse polynomials with $O(\log(n))$-local queries under what they defined as *locally smooth distributions*[1], which in some sense generalize the uniform and product distributions. They also proposed an algorithm that learns DNF formulas under the uniform distribution in quasi-polynomial time using only $O(\log(n))$-local queries.

The exciting ideas of Awasthi et al. (2012) leave many directions for future work. One issue is that their analysis holds for a restricted family of distributions. While these results

[1] locally α-smooth distributions can be defined as the class of distributions for which the logarithm of the density function is $\log(\alpha)$-Lipschitz with respect to the Hamming distance.

provide evidence of the excessive power of local queries, the distributional assumptions are rather strong.

Our work follows Awasthi et al., and is focused on 1-local queries, which are the closest to the original PAC model. We formulate an arguably natural distributional assumption, and present an algorithm that uses 1-local membership queries to learn DNF formulas under this assumption. We also provide a matching lower bound: Namely, we prove that learning DNFs under our assumption is hard without the use of queries, assuming that learning decision trees is hard. This is the first example of a natural problem in which 1-local queries are stronger than the vanilla PAC model (it complements the work of Awasthi et al. who showed a similar result for a highly artificial problem).

Finally, we provide some empirical evidence that using local queries can be helpful in practice, and importantly, that the implementation of the queries is easy, straightforward, and can be acquired by crowdsourcing without the use of an expert. We present a method for using local queries to perform a user-induced feature selection process, and present results of this protocol on the task of sentiment analysis of tweets. Our results show that by acquiring a more expressive data set, using (a variant of) 1-local queries, we can achieve better results with fewer examples. Based on the fact that a smaller data set is sufficient, we gain twice: we need less manpower for the labeling process and less computing power for the training process. We note that similar experiments also present encouraging results along this line (Raghavan and Allan, 2007; Raghavan et al., 2005; Settles, 2011; Druck et al., 2009). This supplies more evidence that such query-based methods can be useful in practice.

2 Previous Work

2.1 PAC

Valiant's Probably Approximately Correct (PAC) model of learning (Valiant, 1984) formulates the problem of learning a concept from examples. Examples are chosen according to a fixed but unknown and arbitrary distribution on the instance space. The learner's task is to find a prediction rule. The requirement is that with high probability, the prediction rule will be correct on all but a small fraction of the instances.

A few positive results are known in this model - i.e., concept classes that have been proven to be PAC-learnable. Maybe the most significant example is the class of halfspaces. More examples include relatively weak classes such as DNFs and CNFs with constantly many terms (Valiant, 1984), and rank k decision trees (Ehrenfeucht and Haussler, 1989) for a constant k.

Despite these positive results, most PAC learning problems are probably intractable. In fact, beyond the results mentioned above, almost no positive results are known. Furthermore, several negative results are known. For example, learning automatons, logarithmic depth circuits, and intersections of polynomially many halfspaces are all intractable, assuming the security of various cryptographic schemes (Kearns and Valiant, 1994; Klivans et al., 2006). In (Daniely et al., 2014; Daniely and Shalev-Shwatz, 2014; Daniely et al., 2013), it is shown that learning DNF formulas, and learning intersections of $\omega(\log(n))$ halfspaces are intractable under the assumption that refuting random k-SAT is hard.

2.2 Membership Queries

The PAC model is a "passive" model in which the learner receives a random data set of examples and their labels and then outputs a classifier. A stronger version would be an active model in which the learner gathers information about the world by asking questions and receiving responses. Several types of active models have been proposed: the Membership Query Synthesis, Stream-Based Selective Sampling, and Pool-Based Sampling (Settles, 2010). Our work is in the area of the "Membership Queries" (MQ) model which was presented in (Valiant, 1984). In this model the learner is allowed to query for the label of any particular example that it chooses (even examples that are not in the given sample).

This model has been shown to be stronger in several scenarios. Some examples of concept classes that have been proven to be PAC-learnable only if membership queries are available include: The class of Deterministic Finite Automatons (Angluin, 1987), the class of k-term DNF for $k = \frac{\log(n)}{\log(\log(n))}$ (Blum and Rudich, 1992), the class of decision trees and k-almost monotone-DNF formulas (Bshouty, 1995), the class of intersections of k-halfspaces (Baum, 1991) and the class of DNF formulas under the uniform distribution (Jackson, 1994). The last of these results was built upon Freund's boosting algorithm (Freund, 1995) and the Fourier-based technique for learning using membership queries due to (Kushilevitz and Mansour, 1993).

It should be noted that there are cases in which the additional strength of MQ does not help. E.g., in the case of learning DNF and CNF formulas (Angluin and Kharitonov, 1995), and in the case of distribution free agnostic learning (although in the distribution-specific agnostic setting membership queries do increase the power of the learner) (Feldman, 2009).

2.3 Baum and Lang

As discussed above, there has been widespread and significant theoretical work in the PAC + MQ model. On the other hand, almost no practical work on implementing these ideas has been done. A well-known exception is the work of Baum and Lang (1992). They applied a variation of the MQ algorithm for learning a linear classifier proposed in Baum (1991). This algorithm uses the idea that given two examples, one positive and one negative, and a query oracle, it is possible to find an approximately accurate separating halfspace by using a binary search on the line between the positive and negative examples. Their experiment attempts to evaluate this idea in practice. The task that they chose is the task of binary digit classification. The algorithm would receive two examples, one positive and one negative (say, an image of the digit 4 and an image of the digit 7) and would return the weights of the halfspace. The generalization error of the halfspace would then be tested on other examples from the data. The query technique they used in the experiment is different than in the original algorithm: "A direct implementation of this algorithm would repeatedly flash images on the screen during the binary search and would require the test subject to type in the correct label for each image. Because this process seemed likely to be error prone, we instead provided an interface that permitted the test subject to scan through the input space using the mouse and then click on an image that seemed to lie right at the edge of recognizability" (from Baum and Lang (1992)).

For an example of what the users saw on the screen see figure 1.

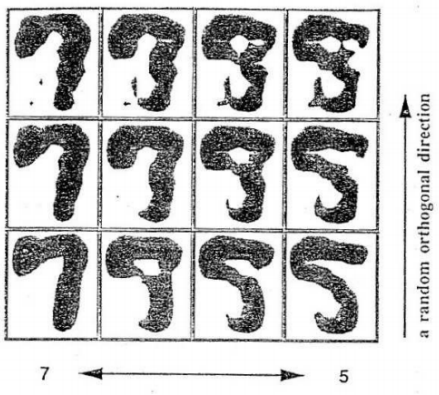


Figure 1: An example taken from (Baum and Lang, 1992): the images the user saw on the screen for the digits 5 and 7

They compared the performance of their algorithm to five other variants, three classic PAC (sample based) algorithms: Backpropogation, Perceptron and simplex, and two baselines: the first returns the perpendicular bisection of the line segments connecting the two examples, and the second returns a randomly oriented hyperplane through the midpoint of the line. The query learning algorithm uses the additional information obtained from the users as described above, while the three PAC algorithms use additional examples drawn from the data set. All three PAC algorithms outperformed the query-based algorithm. More surprisingly, even the baseline of choosing the perpendicular bisection line had significantly better results than the halfspace created by the query algorithm. The only method that

was worse than the query based method was the random bisector method. They suggest that the reason for the poor results is that the question the users had to answer, to find the boundary pattern, lay outside the range of the human competence.

This work led many to the conclusion that membership queries are not useful in practice (Settles (2010); Balcan et al. (2006); Dasgupta (2004) and more). We argue that there are several problems with this conclusion. First and foremost, the task that the users were asked to perform (scanning through images and finding the boundary between digits) is not an intuitive task, and it is very easy to think of other variants for queries which would be more suitable. It is therefore not surprising that the labeling turned out to be noisy considering the nature of the question at hand. Second, their algorithm did not use the PAC abilities; it used queries but did not use the additional option to sample extra points for the data.

2.4 Local Membership Queries

Several suggestions have been made of ways to solve the problem of the algorithm's generation of unnatural examples. The most common one was to drop the whole framework of membership queries and focus on the other types of active learning: stream-based and pool-based. The idea is to filter existing examples taken from a large unlabeled data set drawn from the distribution rather than creating artificial examples. Another suggestion is to give the human annotator the option of answering "I don't know", or to be tolerant of some incorrect answers. The theoretical framework is the model of an *incomplete membership oracle* in which the answers to a random subset of the queries may be missing. This notion was first presented in Angluin and Slonim (1994), and then followed by the notion of *limited* MQ and *malicious* MQ. (Angluin et al. (1997); Blum et al. (1995); Sloan and Turán (1994); Bisht et al. (2008)).

The third method is to restrict the examples that the learning algorithm can query to examples that are similar to examples drawn from the distribution. This is formalized in the work of Awasthi et al. (2012). They present the concept of learning using only *local* membership queries. This framework deals with the problem raised by (Baum and Lang, 1992). By questioning about examples which are close to examples from the distribution we escape the problem of generating random or non-classifiable examples.

The work of Awasthi et al. focused on the n-dimensional boolean hyper-cube $\mathcal{X} = \{-1, 1\}^n$ and on $O(\log(n))$-local queries, i.e., the learning algorithm is given the option to query the label of any point for which there exists a point in the training sample with hamming distance lower than $O(\log(n))$. The model they suggested is a mid-way model between the PAC model (0-local queries) and the PAC + MQ model (n-local queries). Their main result is that t-sparse polynomials are learnable under *locally smooth* distributions using $O\left(\log(n) + \log(t)\right)$-local queries. Another interesting result that they presented is that the class of DNF formulas is learnable under the uniform distribution in quasi-polynomial time $(n^{O(\log\log n)})$ using $O(\log(n))$-local queries. They also presented some results regarding the strength of local MQ. They proved that under standard cryptographic assumptions, using $(r+1)$-local queries is more powerful than using r-local queries (for every $1 \leqslant r \leqslant n-1$). They also showed that local queries do not always help. They showed that if a concept class is agnostically learnable under the uniform distribution using k-local queries

(for constant k) then it is also agnostically learnable (under the uniform distribution) in the PAC model.

2.5 Other Related Work

In section 5, we give some experimental evidence that the use of extra information from the user is helpful. There have been other works along the same line. Druck et al. (2009) propose a pool-based active learning approach in which the user provides labels for input features, rather than instances. The users are asked to provide a "label" for input features, where a labeled input feature denotes that a particular feature is highly indicative of a particular label. Following that, Settles (2011) presented an active learning annotation interface, in which the users label instances and features simultaneously. At any point in time, an instance and a list of features for each label is presented on the screen. The user can choose to either label the instance, choose a feature from the list as being indicative, or add a new feature of his or her choice. Another similar work is of Raghavan and Allan (2007) and Raghavan et al. (2005). They studied the problem of tandem learning where they combine uncertainty sampling for instances along with co-occurrence-based interactive feature selection. All the above experiments were conducted on the text domain and the features were always unigrams. The experiments presented encouraging results of using the human annotators, either by reaching better results, or by showing that the excessive use of annotators can reduce the size of the data set, and sometimes both.

3 Setting

3.1 The PAC Model

Our framework is an extension of the PAC (Probably Approximately Correct) model of learning. Before introducing it, we will briefly review PAC learning. We will only consider binary classification where the instance space is $\mathcal{X} = \mathcal{X}_n = \{-1,1\}^n$ and the label space is $\mathcal{Y} = \{0,1\}$. A learning problem is defined by a hypothesis class $\mathcal{H} \subset \{0,1\}^{\mathcal{X}}$. We assume that the learner receives a *training set*

$$S = \{(\mathbf{x}_1, h^\star(\mathbf{x}_1)), (\mathbf{x}_2, h^\star(\mathbf{x}_2)), \ldots, (\mathbf{x}_m, h^\star(\mathbf{x}_m))\} \in (\mathcal{X} \times \mathcal{Y})^m$$

where the $\mathbf{x}_i$'s are sampled i.i.d. from some *unknown* distribution $\mathcal{D}$ on $\mathcal{X}$ and $h^\star : \mathcal{X} \to \mathcal{Y}$ is some *unknown* hypothesis. We will focus on the so-called realizable case where $h^\star$ is assumed to be in $\mathcal{H}$. The learner returns (a description of) a hypothesis $\hat{h} : \mathcal{X} \to \mathcal{Y}$. The goal is to approximate $h^\star$, namely to find $\hat{h} : \mathcal{X} \to \mathcal{Y}$ with *loss* as small as possible, where the loss is defined as $L_{\mathcal{D},h^\star}(\hat{h}) = \mathbb{P}_{\mathbf{x}\sim\mathcal{D}}\left(\hat{h}(\mathbf{x}) \neq h^\star(\mathbf{x})\right)$. We will require our algorithms to return a hypothesis with loss $< \epsilon$ in time that is polynomial in n and $\frac{1}{\epsilon}$. Concretely,

Definition 1 (Learning algorithm) *We say that a learning algorithm $\mathcal{A}$* **PAC learns** *$\mathcal{H}$ if*

- *There exists a function $m_{\mathcal{A}}(n,\epsilon) \leqslant \text{poly}\left(n, \frac{1}{\epsilon}\right)$, such that for every distribution $\mathcal{D}$ over $\mathcal{X}$, every $h^\star \in \mathcal{H}$ and every $\epsilon > 0$, if $\mathcal{A}$ is given a training sequence*

  $$S = \{(\mathbf{x}_1, h^\star(\mathbf{x}_1)), (\mathbf{x}_2, h^\star(\mathbf{x}_2)), \ldots, (\mathbf{x}_m, h^\star(\mathbf{x}_m))\}$$

  where the $\mathbf{x}_i$'s are sampled i.i.d. from $\mathcal{D}$ and $m \geqslant m_{\mathcal{A}}(n,\epsilon)$, then with probability of at least $\frac{3}{4}$ (over the choice of S)[2], the output $\hat{h}$ of $\mathcal{A}$ satisfies $L_{\mathcal{D},h^\star}(\hat{h}) < \epsilon$.

- *Given a training set of size m*
  - *$\mathcal{A}$ runs in time* $\text{poly}(m,n)$.
  - *The hypothesis returned by $\mathcal{A}$ can be evaluated in time* $\text{poly}(m,n)$.

Definition 2 (PAC learnability) *We say that a hypothesis class $\mathcal{H}$ is* ***PAC learnable*** *if there exists a PAC learning algorithm for this class.*

3.2 (Local) Membership Queries Model

Learning with membership queries is an extension of the PAC model in which the learning algorithm is allowed to *query* the labels of specific examples in the domain set. A membership query is a call to an ORACLE which receives as input some $\mathbf{x} \in \mathcal{X}$ and returns $h^\star(\mathbf{x})$. This is called a "membership query" because the ORACLE returns 1 if $\mathbf{x}$ is in the set of examples positively labeled by $h^\star$.

[2]The success probability can be amplified to $1-\delta$ by repetition.

Definition 3 (Membership-Query Learning Algorithm) *We say that a learning algorithm $\mathcal{A}$* **learns** *$\mathcal{H}$* **with membership queries** *if*

- *There exists a function $m_{\mathcal{A}}(n, \epsilon) \leqslant \text{poly}\left(n, \frac{1}{\epsilon}\right)$, such that for every distribution $\mathcal{D}$ over $\mathcal{X}$, every $h^\star \in \mathcal{H}$ and every $\epsilon > 0$, if $\mathcal{A}$ is given access to* membership queries*, and a training sequence*

$$S = \{(\mathbf{x}_1, h^\star(\mathbf{x}_1)), (\mathbf{x}_2, h^\star(\mathbf{x}_2)), \ldots, (\mathbf{x}_m, h^\star(\mathbf{x}_m))\}$$

  where the $\mathbf{x}_i$'s are sampled i.i.d. from $\mathcal{D}$ and $m \geqslant m_{\mathcal{A}}(n, \epsilon)$, then with probability of at least $\frac{3}{4}$ (over the choice of S), the output $\hat{h}$ of $\mathcal{A}$ satisfies $L_{\mathcal{D},h^\star}(\hat{h}) < \epsilon$.

- *Given a training set of size m*
  - *$\mathcal{A}$ asks at most* $\text{poly}(m, n)$ *membership queries.*
  - *$\mathcal{A}$ runs in time* $\text{poly}(m, n)$.
  - *The hypothesis returned by $\mathcal{A}$ can be evaluated in time* $\text{poly}(m, n)$.

Our work will deal with a specific type of membership queries, ones that are in some way close to examples that are already in the sample. Concretely, we say that a membership query $\mathbf{x} \in \mathcal{X}$ is q-**local** if there exists a training example x' whose Hamming distance[3] from $\mathbf{x}$ is at most q.

Definition 4 (Local-Query Learning Algorithm) *We say that a learning algorithm $\mathcal{A}$* **learns** *$\mathcal{H}$* **with** *q*-**local membership queries** *if $\mathcal{A}$ learns $\mathcal{H}$ with membership queries that are all q-local.*

Definition 5 *We say that a hypothesis class $\mathcal{H}$ is* ***q-LQ learnable*** *if there exists a q-Local-query learning algorithm for this class.*

Learning Under a Specific Family of Distributions

In the classic PAC model discussed above, the learning algorithm needs to be probably-approximately correct for *any distribution* $\mathcal{D}$ on $\mathcal{X}$ and *any hypothesis* $h^\star \in \mathcal{H}$. In this work we will have guarantees with respect to more restricted families. We will say that $\mathcal{A}$ **learns** $\mathcal{H}$ **w.r.t a family** $\mathscr{D}$ of pairs $(\mathcal{D}, h)$ of distributions on $\mathcal{X}$ and hypotheses in $\mathcal{H}$ if the following holds: The algorithm $\mathcal{A}$ satisfies the requirements of a learning algorithm whenever the pair $\mathcal{D}$ and h in the definition of a learning algorithm belongs to $\mathscr{D}$. Similar considerations apply also to the notion of learning with (local) membership queries.

4 Learning DNFs with Evident Examples Using 1-local MQ

4.1 Definitions and Notations

Definition 6 (Disjunction Normal Form Formula) *A* **DNF term** *is a conjunction of literals. A* **DNF formula** *is a disjunction of DNF terms.*

[3] We only consider the instance space $\{-1, 1\}^n$, so the hamming distance is natural. However, the definition can be extended to other metrics.

Each DNF formula over n variables naturally induces a function $h : \{-1,1\}^n \to \{0,1\}$ (when we standardly identify $\{0,1\}$ with "True" and "False"). We denote by h_F the function induced by the DNF formula F.

Remark 1 *We will look at succinctly described hypotheses (e.g., a* DNF *with a small number of terms) and on small, but non-negligible probabilities. For simplicity, we will take the convention that* **small** *is at most* n^2 *and* **non negligible** *is at least* $\frac{1}{n^3}$*. All of our results can be easily generalized to the case where "small" and "non-negligible" are defined as* $\leqslant n^{c_1}$ *and* $\geqslant \frac{1}{n^{c_2}}$ *for any constants* $c_1, c_2 > 0$*.*

Definition 7 *Denote by* $\mathcal{H}_{\mathrm{DNF}}$ *the hypothesis class of all functions that can be realized by a* DNF *with a small number of terms. That is*

$$\mathcal{H}_{\mathrm{DNF}} = \{h_F : \ F \text{ is a DNF formula with at most } n^2 \text{ terms}\}$$

Intuitively, when evaluating a DNF formula on a given example, we check a few conditions (corresponding to the formula's terms), and deem the example positive if one of the conditions holds. We will consider the case that for each of these conditions, there is some chance to see a "prototype example". Namely, an example that satisfies only this condition in a strong (or evident) way.

Definition 8 *Let* $F = T_1 \vee T_2 \vee \ldots \vee T_d$ *be a DNF formula. An example* $\mathbf{x} \in \{-1,1\}^n$ *satisfies a term* T_i *(with respect to the formula* F*)* **evidently** *if :*

- *It satisfies* T_i*. (In particular,* $h_F(\mathbf{x}) = 1$*)*
- *It does* **not** *satisfy any other term* T_k *(for* $k \neq i$*) from* F*.*
- *No coordinate change will turn* T_i *False and another term* T_k *True. Concretely, if for* $j \in [n]$ *we denote* $\mathbf{x}^{\oplus j} = (x_1, \ldots, x_{j-1}, -x_j, x_{j+1}, \ldots, x_n)$*, then for every coordinate* $j \in [n]$*, if* $\mathbf{x}^{\oplus j}$ *satisfies* F *(i.e. if* $h_F(\mathbf{x}^{\oplus j}) = 1$*) then* $\mathbf{x}^{\oplus j}$ *satisfies* T_i *and only* T_i*.*

The first distributional assumption that we consider is that each positive example satisfies one term evidently.

Definition 9 *A pair* $(\mathcal{D}, h^\star)$ *of a distribution* $\mathcal{D}$ *over* $\{-1,1\}^n$ *and* $h^\star : \{-1,1\}^n \to \{0,1\}$ *is* ***realized by a small DNF with evident examples*** *if there exists a DNF formula* $F = T_1 \vee T_2 \vee \ldots \vee T_d$ *over* $\{-1,1\}^n$ *with* $d \leqslant n^2$ *such that* $h^\star = h_F$ *and additionally, every positive example* $\mathbf{x} \in \{-1,1\}^n$ *with* $\mathcal{D}(\mathbf{x}) > 0$ *satisfies one of* F*'s terms evidently.*

One of the assumptions in our definition is that the target function can be realized by a DNF formula for which every example satisfies at most one term. For a function that is realized by a decision tree this always holds. So, in a sense, our assumption holds for functions that can be realized by a "stable" decision tree.

The above definition makes a strong assumption, namely that *every* positive example is an evidence for one term. The next definition relaxes that assumption and only assumes that for every term there is a non-negligible probability to see an evident example.

Definition 10 *A pair $(\mathcal{D}, h^\star)$ of a distribution $\mathcal{D}$ over $\{-1,1\}^n$ and $h^\star : \{-1,1\}^n \to \{0,1\}$ is* ***weakly realized by a small*** DNF ***with evident examples*** *if there exists a DNF formula $F = T_1 \vee T_2 \vee \ldots \vee T_d$ over $\{-1,1\}^n$ with $d \leqslant n^2$ such that $h^\star = h_F$ and for every term T_i there is a non-negligible[4] probability to see an example that satisfies this term evidently.*

For example, our assumption holds for every distribution $\mathcal{D}$, provided that $h^\star$ can be realized by a DNF formulas in which any pair of different terms contains two opposite literals.

4.2 Upper Bounds

We will now present two learning algorithms that use 1-LQ, and prove that each of these algorithms learn the class $\mathcal{H}_{\text{DNF}}$ with respect to the families of distributions defined above. Both algorithms use the following claim that follows directly from definition 8

Claim 1 *Let $F = T_1 \vee T_2 \vee \ldots \vee T_d$ be a* DNF *formula over $\{-1,1\}^n$. Then for every $\mathbf{x} \in \{-1,1\}^n$ that satisfies a term T_i evidently (with respect to F), for every $j \in [n]$ it holds that:*

$$h_F(\mathbf{x}^{\oplus j}) = 1 \iff \textit{the term } T_i \textit{ does not contain the variable } x_j$$

Algorithm 1 Create a DNF formula

Input: $S \in (\{-1,1\}^n \times \{0,1\})^m$
Output: A DNF formula H

```
start with an empty DNF formula H
for all (x, y) ∈ S do
  if y = 1 then
    define T = x_1 ∧ x̄_1 ∧ x_2 ∧ x̄_2 ∧ ... ∧ x_n ∧ x̄_n
    for 1 ⩽ j ⩽ n do
      query x^{⊕j} (to get h*(x^{⊕j}))
      if h*(x^{⊕j}) = 1 then
        remove x_j and x̄_j from T
      if h*(x^{⊕j}) = 0 then
        if x_j = 1 then
          remove x̄_j from T
        if x_j = 0 then
          remove x_j from T
  H = H ∨ T
return H
```

Theorem 1 *The hypothesis class $\mathcal{H}_{\text{DNF}}$ is 1-LQ learnable with respect to distributions that are realized by a* DNF *with evident examples.*

[4] Recall that non-negligible is at least $\frac{1}{n^3}$

Proof We will prove that algorithm 1 learns $\mathcal{H}_{\text{DNF}}$ with 1-local membership queries. First, it is easy to see that this algorithm is efficient: For a training set of size m the algorithm asks for at most $n \cdot m$ 1-local membership queries, and runs in time $O(nm)$. Likewise, the hypothesis that the algorithm returns is a DNF formula with at most m terms and every term is of size at most n, therefore it can be evaluated in time polynomial in mn.

Now, let $\mathcal{D}$ be a distribution on $\{-1,1\}^n$ and $h^\star : \{-1,1\}^n \to \{0,1\}$ be a hypothesis such that the pair $(\mathcal{D}, h^\star)$ is realized by a small DNF with evident examples. Let $F = T_1 \vee T_2 \vee \ldots \vee T_d$ be that small DNF formula, (in particular $h^\star = h_F$ and $d \leqslant n^2$). For $\epsilon > 0$ we take a sample $S = \{(\mathbf{x}_i, h^\star(\mathbf{x}_i)\}_{i=1}^m$ where $\{\mathbf{x}_i\}_{i=1}^m$ are sampled i.i.d from $\mathcal{D}$ and $m = \frac{2n^2}{\epsilon} \log \frac{2n^2}{\epsilon} \geqslant \frac{2d}{\epsilon} \log \frac{2d}{\epsilon}$.

Let H be the DNF formula returned by the algorithm after running on S, and let $\hat{h}$ be the function induced by H. We will prove that with probability of at least 3/4 (over the choice of the examples) $L_{\mathcal{D},h^\star}(\hat{h}) < 4\epsilon$.

From the assumption on the distribution we get that every instance $\mathbf{x}$ that satisfies the formula (in our case every $\mathbf{x}$ such that $(\mathbf{x}, 1) \in S$), satisfies exactly one term T. For every one of these positive instances from S, we will show that we add that exact term to H. For every such $\mathbf{x}$ we start with a full term (containing all the possible literals) and then for every $j \in [n]$, at iteration j:

- if $h^\star(\mathbf{x}) = h^\star(\mathbf{x}^{\oplus j}) = 1$ we know from claim 1 that the variable x_j cannot appear in T - so we remove it and its negation from the current term.
- if $h^\star(\mathbf{x}) = 1$ and $h^\star(\mathbf{x}^{\oplus j}) = 0$ we know that either x_j or $\overline{x_j}$ appears in T and we remove the one that cannot appear in T according to the value of x_j.

After n iterations we get exactly T - the term that $\mathbf{x}$ satisfies evidently. Therefore - H will contain every term from F for which there was an instance $\mathbf{x}$ in S that satisfies it - other then that H will contain no other terms. In other words,

$$\mathbb{P}_{\mathbf{x}\sim\mathcal{D}}[h^\star(\mathbf{x}) = 0 \wedge \hat{h}(\mathbf{x}) = 1] = 0$$

and we get that

$$L_{\mathcal{D},h^\star}(\hat{h}) = \mathbb{P}_{\mathbf{x}\sim\mathcal{D}}[h^\star(\mathbf{x}) \neq \hat{h}(\mathbf{x})] = \mathbb{P}_{\mathbf{x}\sim\mathcal{D}}[h^\star(\mathbf{x}) = 1 \wedge \hat{h}(\mathbf{x}) = 0]$$

Denote by p_i the probability to sample $\mathbf{x}$ (from $\mathcal{D}$) that will satisfy T_i, and let A_i be the event that S did not contain any $\mathbf{x}$ which satisfies T_i. Then

$$\begin{aligned}\mathbb{P}_{\mathbf{x}\sim\mathcal{D}}[h^\star(\mathbf{x}) = 1 \wedge \hat{h}(\mathbf{x}) = 0] &= \mathbb{P}_{\mathbf{x}\sim D}[\exists i \in [d] \text{ such that } \mathbf{x} \text{ satisfies } T_i \wedge \hat{h}(\mathbf{x}) = 0] \\ &\leqslant \sum_{i=1}^{d} \mathbb{P}_{\mathbf{x}\sim\mathcal{D}}[\mathbf{x} \text{ satisfies } T_i \wedge \hat{h}(\mathbf{x}) = 0] = \sum_{i=1}^{d} p_i \cdot \mathbb{1}_{A_i}\end{aligned}$$

Notice that since p_i is the probability to sample x we get that $\mathbb{P}_{S\sim D^m}[A_i] = (1-p_i)^m$

Now if we look at the expectation we get

$$\begin{aligned}
\mathop{\mathbb{E}}_{S\sim\mathcal{D}^m}[L_{\mathcal{D},h^\star}(\hat{h})] &\leqslant \mathop{\mathbb{E}}_{S\sim\mathcal{D}^m}[\sum_{i=1}^{d} p_i \cdot \mathbb{1}_{A_i}] \\
&= \sum_{i=1}^{d} p_i \mathop{\mathbb{E}}_{S\sim D^m}[\mathbb{1}_{A_i}] \\
&= \sum_{i=1}^{d} p_i \mathop{\mathbb{P}}_{S\sim D^m}[A_i] \\
&= \sum_{i=1}^{d} p_i (1-p_i)^m \\
&= \sum_{i|p_i<\frac{\epsilon}{2d}} p_i(1-p_i)^m + \sum_{i|p_i\geqslant\frac{\epsilon}{2d}} p_i(1-p_i)^m \\
&\leqslant \sum_{i|p_i<\frac{\epsilon}{2d}} \frac{\epsilon}{2d} + \sum_{i|p_i\geqslant\frac{\epsilon}{2d}} (1-p_i)^m \\
&\leqslant d\cdot\frac{\epsilon}{2d} + \sum_{i|p_i\geqslant\frac{\epsilon}{2d}} e^{-mp_i} \\
&\leqslant \frac{\epsilon}{2} + d\cdot e^{-m\frac{\epsilon}{2d}}
\end{aligned}$$

Since $m \geqslant \frac{2d}{\epsilon}\log\frac{2d}{\epsilon}$ we get $\mathbb{E}[L_{\mathcal{D},h^\star}(\hat{h})] < \epsilon$ and using Markov's inequality we obtain

$$\mathop{\mathbb{P}}_{S\sim D^m}[L_{\mathcal{D},h^\star}(\hat{h})] \geqslant 4\epsilon] \leqslant \frac{\mathbb{E}[L_{\mathcal{D},h^\star}(\hat{h})]}{4\epsilon} < \frac{1}{4}$$

■

Algorithm 2 Create a DNF formula with checking and deleting false terms

Input: $S_1, S_2 \subseteq (\{-1,1\}^n \times \{-1,1\})^m$
Output: a DNF formula H

```
start with an empty DNF formula H
for all (x, y) ∈ S_1 do
  if y = 1 then
    define T = x_1 ∧ x̄_1 ∧ x_2 ∧ x̄_2 ∧ ... ∧ x_n ∧ x̄_n
    for 1 ⩽ j ⩽ n do
      query x^{⊕j} (to get h*(x^{⊕j}))
      if h*(x^{⊕j}) = 1 then
        remove x_j and x̄_j from T
      if h*(x^{⊕j}) = 0 then
        if x_j = 1 then
          remove x̄_j from T
        if x_j = 0 then
          remove x_j from T
    H = H ∨ T
  for all T in H do
    for all (x, y) ∈ S_2 do
      if T(x) = 1 but y = 0 then
        remove T from H
return H
```

Theorem 2 *The hypothesis class $\mathcal{H}_{\text{DNF}}$ is 1-LQ learnable with respect to distributions that are weakly realized by a* DNF *with evident examples.*

Proof We will prove that algorithm 2 learns $\mathcal{H}_{\text{DNF}}$ with 1-local membership queries. In this case we will have two sample sets - S_1 of size m_1 which will be used as before - to build the terms of H, and S_2 of size m_2 - a separate set to check the terms that were built. Again, it is easy to see that this algorithm is efficient. For training sets S_1 of size m_1 and S_2 of size m_2 the algorithm asks for at most $n \cdot m_1$ 1-local membership queries. The running time of the first loop is $O(nm_1)$ and in that loop we add at most m_1 terms to H so the running time of the second loop is $O(m_1 m_2)$. All in all the running time is polynomial in (m_1, m_2, n). Also, the hypothesis that the algorithm returns is a DNF formula with at most m_1 terms and every term is of size at most n, therefore it can be evaluated at time polynomial in $m_1 n$.

Now, let $\mathcal{D}$ be a distribution on $\{-1,1\}^n$ and $h^\star : \{-1,1\}^n \to \{0,1\}$ be a hypothesis such that the pair $(\mathcal{D}, h^\star)$ is realized by a small DNF with evident examples. Let $F = T_1 \vee T_2 \vee \ldots \vee T_d$ be that small DNF formula, (in particular $h^\star = h_F$ and $d \leqslant n^2$). Denote by $H = \hat{T}_1 \vee \hat{T}_2 \vee \ldots \vee \hat{T}_k$ the DNF formula algorithm 2 returns. Following the same argument from the last proof, a term T_i will be added to H in the first loop if S_1 contains an example that satisfies T_i evidently. We will define m_1 so that with high probability for every term T_i there will be $(\mathbf{x}, 1) \in S_1$ such that $\mathbf{x}$ satisfies T_i evidently.
Denote by s_i the probability to sample $\mathbf{x}$ (from $\mathcal{D}$) that satisfies T_i evidently, and let

$s = \min\{s_i\}_{i=1}^{d}$. Since for every term the probability to see an evident example is non-negligible, $s \geqslant n^{-3}$. For every i, the probability of *not* seeing an example in S_1 that satisfies T_i evidently is

$$(1-s_i)^m \leqslant (1-s)^m \leqslant e^{-sm} \leqslant e^{-\frac{m}{n^3}}$$

If we set m_1 to be $n^3 \log(8n^2) \geqslant n^3 \log(8d)$ we get that the probability of not seeing an example that satisfies T_i evidently (when sampling S_1 from $\mathcal{D}^{m_1}$) is less than $\frac{1}{8d}$ and from the union bound we get that the probability that the sample will contain an evident example for every term is at least $\frac{7}{8}$. Therefore with probability of at least $\frac{7}{8}$ we will add every T_i to H in the first loop. In the second loop, when we remove terms from H, we only remove terms which contradicts one of the examples in S_2. Since all of the examples in the sample set are labeled by F, we will never remove a term that is a part of F Therefore with probability of at least $\frac{7}{8}$ H will contain *all* of F's terms. Formally,

$$\mathop{\mathbb{P}}_{S_1 \sim \mathcal{D}^{m_1}}\Big[\mathop{\mathbb{P}}_{\mathbf{x}\sim\mathcal{D}}[h^\star(\mathbf{x}) = 1 \wedge \hat{h}(\mathbf{x}) = 0] = 0\Big] \geqslant \frac{7}{8}$$

Note that we are not done, as the algorithm might create a wrong term (when using a "non-evident" example). For this reason we add the second loop. We use the sample S_2 to test every term $\hat{T}_i$ that was added to H in the first loop. If we see an example $\mathbf{x}$ such that $\hat{T}_i(\mathbf{x}) = 1$ but $h^\star(\mathbf{x}) = 0$ we remove $\hat{T}_i$ and continue to the next term. Now denote by p_i the probability to sample $\mathbf{x}$ (from $\mathcal{D}$) that will satisfy $\hat{T}_i$, and by A_i the event that $\hat{T}_i$ is a wrong term (not from F) but the "checking" step did not discover that. Then

$$\begin{aligned}
\mathop{\mathbb{P}}_{\mathbf{x}\sim\mathcal{D}}[\hat{h}(\mathbf{x}) = 1 \wedge h^\star(\mathbf{x}) = 0] &= \mathop{\mathbb{P}}_{\mathbf{x}\sim\mathcal{D}}[\exists i \in [k] \text{ such that } \mathbf{x} \text{ satisfies } \hat{T}_i \wedge h^\star(\mathbf{x}) = 0] \\
&\leqslant \sum_{i=1}^{k} \mathop{\mathbb{P}}_{\mathbf{x}\sim\mathcal{D}}[\mathbf{x} \text{ satisfies } \hat{T}_i \wedge h^\star(\mathbf{x}) = 0] \\
&= \sum_{i=1}^{k} p_i \cdot \mathbb{1}_{A_i}
\end{aligned}$$

Note that since A_i is the event that there wasn't any example in S_2 which satisfied $\hat{T}_i$ (otherwise the checking step would discover that $\hat{T}_i$ is wrong) this is the same situation as in the proof of theorem 1, so

$$\mathop{\mathbb{P}}_{S_2\sim\mathcal{D}^{m_2}}[A_i] = (1-p_i)^{m_2}$$

By the same analysis of the former proof, we get that if the size of S_2 is $\geqslant \frac{2k}{\epsilon}\log\frac{2k}{\epsilon}$ then

$$\mathop{\mathbb{P}}_{S_2\sim\mathcal{D}^{m_2}}\Big[\mathop{\mathbb{P}}_{\mathbf{x}\sim\mathcal{D}}[h^\star(\mathbf{x}) = 0 \wedge \hat{h}(\mathbf{x}) = 1] \geqslant 4\epsilon\Big] \leqslant \frac{1}{4}$$

Finally we notice that $k \leqslant m_1$, because for each example in S_1 the algorithm adds at most one term to H. So we can set m_1 as above and $m_2 = \frac{2m_1}{\epsilon}\log\frac{2m_1}{\epsilon}$ and if we run algorithm

2 on S_1 and S_2 we get that with probability of at least $1 - (\frac{1}{4} + \frac{1}{8}) = \frac{3}{4} - \frac{1}{8}$ over sampling S_1 and S_2

$$\begin{aligned} L_{\mathcal{D},h^\star}(\hat{h}) &= \mathbb{P}_{\mathbf{x}\sim\mathcal{D}}[h^\star(\mathbf{x}) \neq \hat{h}(\mathbf{x})] \\ &= \mathbb{P}_{\mathbf{x}\sim\mathcal{D}}[h^\star(\mathbf{x}) = 1 \wedge \hat{h}(\mathbf{x}) = 0] + \mathbb{P}_{\mathbf{x}\sim\mathcal{D}}[h^\star(\mathbf{x}) = 0 \wedge \hat{h}(\mathbf{x}) = 1] \\ &\leqslant 0 + 4\epsilon = 4\epsilon \end{aligned}$$

■

4.3 A Lower Bound

In this section we provide evidence that the use of queries in our upper bounds is crucial. We will show that the problem of learning poly-sized decision trees can be reduced to the problem of learning DNFs w.r.t. distributions that are realized by a small DNF with evident examples. As learning decision trees is widely believed to be intractable (in fact, even learning the much smaller class of $\log(n)$-juntas is conjectured to be hard), this reduction serves as an indication that the problems we considered are hard without membership queries.

Definition 11 *A decision tree over $\{-1,1\}^n$ is a binary tree with labels chosen from $x_1, \ldots, x_n$ on the internal nodes, and labels from $\{0,1\}$ on the leaves. Each internal node's left branch is viewed as the -1 branch; the right branch is the 1 branch. Each decision tree over n variables induces a function $h : \{-1,1\}^n \to \{0,1\}$ in the following way: For a decision tree T , a vector $\boldsymbol{a} \in \{-1,1\}^n$ defines a path in the tree from the root to a specific leaf by choosing a_i's branch at each node x_i and the value that the function h_T returns on $\boldsymbol{a}$ is defined to be the label of the leaf at the end of this path.*

Definition 12 *Denote by $\mathcal{H}_{\mathrm{DT}}$ the hypothesis class of all functions that can be realized by a decision tree with a small number of leaves. That is*

$$\mathcal{H}_{\mathrm{DT}} = \{h_T : \ T \text{ is a DT with at most } n^2 \text{ leaves}\}$$

Theorem 3 *PAC learning the hypothesis class $\mathcal{H}_{\mathrm{DNF}}$ w.r.t distributions that are realized by a small* DNF *with evident examples is as hard as PAC learning $\mathcal{H}_{\mathrm{DT}}$.*

The proof will follow from the following claim:

Claim 2 *There exists a mapping (a reduction) $\varphi : \{-1,1\}^n \to \{-1,1\}^{2n}$, that can be evaluated in poly(n) time so that for every decision tree T over $\{-1,1\}^n$ there exists a* DNF *formula F over $\{-1,1\}^{2n}$ such that the following holds:*

1. *The number of terms in F is upper bounded by the number of leaves in T*
2. *$h_T = h_F \circ \varphi$*
3. *$\forall \mathbf{x}$ such that $h_T(\mathbf{x}) = 1$, $\varphi(\mathbf{x})$ satisfies some term in F evidently.*

Proof We will denote $\{-1,1\}^n$ by $\mathcal{X}_n$ and $\{-1,1\}^{2n}$ by $\mathcal{X}_{2n}$.

Define φ as follows:

$$\forall \mathbf{x} = (x_1, x_2, \ldots, x_n) \in \mathcal{X}_n \qquad \varphi(x_1, x_2, \ldots, x_n) = (x_1, x_1, x_2, x_2, \ldots, x_n, x_n)$$

Now, for every tree T, we will build the desired DNF formula F as follows: First we build F' - a DNF formula over $\{-1,1\}^n$. Every leaf labeled '1' in T will define the following term- take the path from the root to that leaf and form the logical AND of the literals describing the path. F' will be a disjunction of these terms. Now, for every term T in F' we will define a term $\phi(T)$ over $\mathcal{X}_{2n}$ in the following way: Let $P_T = \{i \in [n] : x_i \textit{ appear in } T\}$ and $N_T = \{i \in [n] : \overline{x_i} \textit{ appear in } T\}$. So

$$T = \bigwedge_{j \in P_T} x_j \bigwedge_{j \in N_T} \overline{x_j}$$

Define

$$\phi(T) = \bigwedge_{j \in P_T} x_{2j-1} \bigwedge_{j \in P_T} x_{2j} \bigwedge_{j \in N_T} \overline{x_{2j-1}} \bigwedge_{j \in N_T} \overline{x_{2j}}$$

Finally, define F to be the DNF formula over $\mathcal{X}_{2n}$ by

$$F = \bigvee_{T \in F'} \phi(T)$$

We will now prove that φ and F satisfy the required conditions. First, φ can be evaluated in linear time in n. Second, it is easy to see that $h_T = h_F \circ \varphi$, and as every term in F matches one of T's leaves, the number of terms in F cannot exceed the number of leaves in T. It is left to show that the third requirement holds. Let there be an $\mathbf{x}$ such that $h_T(\mathbf{x}) = 1$, then x is matched to one and only one path from T's root to a leaf labeled '1'. From the construction of F, $\mathbf{x}$ satisfies one and only one term in F' because every term is matched to exactly one path from T's root to a leaf labeled 1. Regarding the last requirement - that no coordinate change will make one term from F False and another one True - we made sure this will not happen by "doubling" each variable. By this construction, in order to change a term from False to True at least two coordinate must change their value.

■

Proof [of theorem 3] Suppose we have an efficient algorithm $\mathcal{A}$ that PAC learns $\mathcal{H}_{\mathrm{DNF}}$ with respect to distributions that are realized by DNF with evident examples. Using the reduction from claim 2 we will build an efficient algorithm $\mathcal{B}$ that will PAC learn $\mathcal{H}_{\mathrm{DT}}$. For every training set with examples from $\mathcal{X}_n$:

$$S = \{(\mathbf{x}_1, h^\star(\mathbf{x}_1)), (\mathbf{x}_2, h^\star(\mathbf{x}_2)), \ldots, (\mathbf{x}_m, h^\star(\mathbf{x}_m))\} \in (\mathcal{X}_n \times \{0,1\})^m$$

we define a matching training set with examples from $\mathcal{X}_{2n}$, using φ from the above claim:

$$\tilde{S} := \{(\varphi(\mathbf{x}_1), h^\star(\mathbf{x}_1)), (\varphi(\mathbf{x}_2)), h^\star(\mathbf{x}_2)), \ldots, (\varphi(\mathbf{x}_m), h^\star(\mathbf{x}_m))\} \in (\mathcal{X}_{2n} \times \{0,1\})^m$$

The algorithm $\mathcal{B}$ will work as follows:
Given a training set S, $\mathcal{B}$ will construct $\tilde{S} = \varphi(S)$ and then run $\mathcal{A}$ with input $\tilde{S}$. Let $\hat{h}$ be the output of $\mathcal{A}$ when running on $\tilde{S}$, $\mathcal{B}$ will return $\hat{h} \circ \varphi$. Since φ can be evaluated in $poly(n)$ time and $\mathcal{A}$ is efficient, we get that $\mathcal{B}$ is also efficient.

We will prove that algorithm $\mathcal{B}$ is a learning algorithm for the class $\mathcal{H}_{\mathrm{DT}}$. Since $\mathcal{A}$ is a learning algorithm for the class $\mathcal{H}_{\mathrm{DNF}}$ with respect to distributions that are realized by a small DNF with evident examples, there exists a function $m_{\mathcal{A}}(n, \epsilon) \leqslant \mathrm{poly}\left(n, \frac{1}{\epsilon}\right)$, such that for every $(\mathcal{D}, h^\star)$ that is realized by a small DNF with evident examples and every $\epsilon > 0$, if $\mathcal{A}$ is given a training sequence

$$S = \{(\mathbf{x}_1, h^\star(\mathbf{x}_1)), (\mathbf{x}_2, h^\star(\mathbf{x}_2)), \ldots, (\mathbf{x}_m, h^\star(\mathbf{x}_m))\}$$

where the $\mathbf{x}_i$'s are sampled i.i.d. from $\mathcal{D}$ and $m \geqslant m_{\mathcal{A}}(n, \epsilon)$, then with probability of at least $\frac{3}{4}$ (over the choice of S), the output $\hat{h}$ of $\mathcal{A}$ satisfies $L_{\mathcal{D},h^\star}(\hat{h}) \leqslant \epsilon$.

Let $\mathcal{D}$ be a distribution on $\mathcal{X}_n$ and let h_T be a hypothesis that can be realized by a small DT. Define a distribution $\tilde{\mathcal{D}}$ on $\mathcal{X}_{2n}$ by,

$$(\tilde{\mathcal{D}})(\mathbf{z}) = \begin{cases} \mathcal{D}(\mathbf{x}) & \text{if } \exists \mathbf{x} \in \mathcal{X}_n \text{ such that } \mathbf{z} = \varphi(\mathbf{x}) \\ 0 & \text{otherwise} \end{cases}$$

Since φ is one-to-one, $\tilde{\mathcal{D}}$ is well defined and is a valid distribution on $\mathcal{X}_{2n}$.

Now, as h_T is realized by a small DT, then from the conditions that φ satisfies we get that there exists a DNF formula F such that $h_T = h_F \circ \varphi$ and the pair $(\tilde{\mathcal{D}}, h_F)$ is realized by a small DNF with evident examples. Now for every $\epsilon > 0$ we take a sample $S = \{(\mathbf{x}_1, h_T(\mathbf{x}_1)), (\mathbf{x}_2, h_T(\mathbf{x}_2)), \ldots, (\mathbf{x}_m, h_T(\mathbf{x}_m))\}$ with $m = m_{\mathcal{A}}(2n, \epsilon)$ and obtain that with probability of at least $\frac{3}{4}$ it holds that

$$\begin{aligned} L_{\mathcal{D},h_T}(\mathcal{B}(S)) &= L_{\mathcal{D},h_T}(\hat{h} \circ \varphi) \\ &= \mathop{\mathbb{P}}_{\mathbf{x}\sim\mathcal{D}}[h_T(\mathbf{x}) \neq \hat{h} \circ \varphi(\mathbf{x})] \\ &= \mathop{\mathbb{P}}_{\mathbf{x}\sim\mathcal{D}}[h_F \circ \varphi(\mathbf{x}) \neq \hat{h} \circ \varphi(\mathbf{x})] \\ &= \mathop{\mathbb{P}}_{\mathbf{z}\sim\tilde{\mathcal{D}}}[h_F(\mathbf{z}) \neq \hat{h}(\mathbf{z})] \\ &= L_{\tilde{\mathcal{D}},h_F}(\hat{h}) \\ &= L_{\tilde{\mathcal{D}},h_F}(\mathcal{A}(\tilde{S})) < \epsilon \end{aligned}$$

So $\mathcal{B}$ is indeed a learning algorithm for the class $\mathcal{H}_{\mathrm{DT}}$

■

5 Experiments

Membership queries are a mean by which we can use human knowledge for improving performance in learning tasks. Human beings have a very rich knowledge and understanding of many problems that the ML community works on. They can provide much more information than merely the category of the object or an answer to a "yes" or "no" question. This knowledge is often basic, and can be acquired without the use of an expert (e.g., using crowd-sourcing). In this section we will present empirical results of an algorithm which takes advantage of this extensive knowledge in order to perform smart feature selection.

In standard supervised classification tasks the user is only asked to give the label of each example. What we did in this task, is to ask for additional information. Specifically, we faced a situation where we had a large number of features, and that these features had an interpretation that is easily understood. For every example in the sample set, we asked the user for its label and *in addition*, we asked which features indicate that this instance is labeled as such. After we finished iterating over the entire sample, we used the information on the relevant features to narrow down the feature space. Concretely, we trained linear classifiers only on the features that were chosen to be indicative by the users.

Arguably, this algorithm gathers additional information in a manner that is similar to using 1-local membership queries. 1-local query tests whether changing the value of a single feature changes the label. This can be seen as asking whether this feature is relevant to the prediction or not. In the algorithm presented here, we ask for the relevant features in a broader way. Namely, we explicitly ask which words are relevant to the corresponding label.

5.1 Is the additional data useful?

When humans make decisions, it is often by very complex thought processes and we do not know whether we can access specific considerations that were used in the decision making process. The first goal of this experiment is to show that at least for some tasks, important parts of this thought process are easily accessible. I.e., that the annotators' knowledge can be retrieved by asking simple questions. The second goal is to show that using this extra knowledge can help significantly decrease the number of tagged examples that are required.

We will formulate the above goals using the notion of *error decomposition*. Let $\hat{h}$ be the classifier returned by the algorithm. We decompose $L_{\mathcal{D}}(\hat{h})$ as a sum of the *approximation error* (the error of the best linear classifier) and the *estimation error* (the difference between $L_{\mathcal{D}}(\hat{h})$ and the approximation error):

$$L_{\mathcal{D}}(h_S) = \epsilon_{app} + \epsilon_{est} \quad \text{where} \quad \epsilon_{app} = \min_{h \in \mathcal{H}} L_{\mathcal{D}}(h) \quad \text{and} \quad \epsilon_{est} = L_{\mathcal{D}}(h_S) - \min_{h \in \mathcal{H}} L_{\mathcal{D}}(h)$$

The approximation error ϵ_{app} measures how good is the class of linear classifiers that we restrict ourselves to. In other words, since the class is linear, how informative are the features we use. The estimation error measures to which extent the algorithm overfits the data.

We can now formulate the above goals into claims on the approximation and estimation error. By applying the *user induced* feature selection mentioned above we can only increase the approximation error, as we reduce the hypothesis class to a smaller one. We will want

to show that the feature space chosen by the users is still expressive enough, so that the increase in the approximation error will be minor. In addition, we will show that the feature selection is effective in the sense that the estimation error decreases significantly.

5.2 Experimental setup

5.2.1 Sentiment Analysis

Sentiment analysis (SA) is the Natural Language Processing task of identifying the attitude of a given text (usually whether it is positive, neutral or negative). This task has been studied in the NLP community for many years at different scale levels. It started off from being a document level classification task (Pang and Lee, 2004), and then the focus shifted to handling the sentence level (Hu and Liu, 2004; Kim and Hovy, 2004). The newest focus is sentiment analysis of Microblog data like Twitter. Working with these informal text genres, on which users post their opnions, emotions, and recations about practically everything, presents new challenges for natural language processing beyond those encountered when working with more traditional text genres such as news-wire or product reviews. Indeed, classical approaches to Sentiment Analysis (Pang and Lee, 2008) are not directly applicable to tweets. While most of them focus on relatively large texts, e.g. movie or product reviews, tweets are very short and fine-grained. Nevertheless, the great prominence of Social Media during the last few years encouraged a focus on the sentiment detection over a microblogging domain. There has been a lot of recent work on sentiment analysis of twitter data. Some examples are (Pak and Paroubek, 2010; Kouloumpis et al., 2011; Davidov et al., 2010; Barbosa and Feng, 2010).

We chose this task to demonstrate our method since each example (tweet) is constructed from a limited number of features (words), making each of these features very important for classification. Therefore, it seems that information supplied by users, can be useful in focusing our attention on the important features. Secondly, if in fact the two claims above hold, it will enable us to use a smaller data set, which is very important for this kind of tasks, since SA (and many more NLP tasks) require a large labeled data set which is often costly.

5.2.2 Dataset

	Negative	Neutral	Positive	All
Train	1234	4193	3012	8439
Test	640	1962	2099	4701

Table 1: The SemEval dataset

We worked with the data set from SemEval (Nakov et al., 2013), a shared task for Sentiment Analysis of Tweets . This dataset is constructed of 13,140 (8,439 train+development and 4,701 test, see Table 1) tweets which were collected over a one-year period spanning from January 2012 to January 2013. The tweets were labeled using the crowd sourcing tool Amazon Mechanical Turk and the labels were filtered to get rid of spammers.

For each sentence (tweet), the users were asked to indicate the overall sentiment of the sentence - positive, negative or neutral [5] and also to mark all the subjective (positive or negative) words/phrases in the sentence[6]. The learning task that we worked on is classifying the sentiment of the entire sentence. Although we only want to predict the sentiment of the tweet, we use these two labellings to get one "richer" labelled data-set. I.e., each instance in our training set holds additional information to its sentiment - which words/phrases in the sentence indicate a positive or negative sentiment.

5.2.3 Pre-processing

Beside simple text, tweets may contain URL addresses, references to other Twitter users (appear as @<username>) or content tags (also called hashtags) assigned by the tweeter (# <tag>). During preprocessing, we performed the following standard manipulations:

- Words were switched to lower case and punctuation marks were removed (apart from a fixed set of smileys)
- Every hyperlink was replaced by the meta-word URL
- Every word starting with @, i.e. a username in twitter syntax, was replaced by the meta-word USR.
- The hashtag sign $'\#'$ was removed from every tag to get a simple word. For example *#perfect* was changed to *perfect*.

5.2.4 Language Model

We used the simple bag-of-words language models of n-grams (in our case unigrams, bigrams and trigrams). I.e., each tweet is represented as a sparse vector in $\{0,1\}^d$, where d is the size of the dictionary and the i'th coordinate equals 1 if and only if the i'th word in the dictionary appears in the tweet. We performed a standard cut-off of rare n-grams [7].

5.2.5 Scoring

The results were evaluated on averaged $F1$ scores. This scoring function is used in the SemEval shared task, and overall a very common scoring function for NLP tasks. The $F1$ score is the harmonic mean of Precision and Recall. Every label has it's $F1$ score. For the positive label, the Precision is the number of tweets that were correctly labeled as positive divided by the total number of tweets that were labeled as positive:

$$P_{POS} = \frac{TP}{TP + FP}$$

[5] The original labeling had 4 classes-[objective, positive, negative, or neutral] but since the turkers tended to mix up between the objective and neutral, the two classes were combined in the final task.

[6] This labelling procedure was originally intended to be used for two separate tasks. The first is, when given a tweet containing a marked instance of a word or a phrase, to identify the sentiment of that instance (i.e., whether the word is negative or positive). The second is identifying the sentiment of the whole tweet (without using the marked words).

[7] without performing this cutoff, the results for the non-query variant are much worse

The Recall of the positive label is the number of tweets that were correctly labeled as positive divided by the total number of positive tweets in the data:

$$R_{POS} = \frac{TP}{TP + FN}$$

The positive label $F1$-score is computed as follows:

$$F_{POS} = 2\frac{P_{POS} \cdot R_{POS}}{P_{POS} + R_{POS}}$$

The negative label $F1$-score F_{NEG} is computed similarly. The final score that the results are evaluated on is the average of the above two:

$$F1 = \frac{1}{2}(F_{POS} + F_{NEG})$$

5.2.6 The algorithm

We compare two variants for the feature space: using the entire feature space (after cutting off the rare n-grams), and using the "query acquired" feature space which contains only features that were selected by the users as positive or negative for some example. Information about the data and the number of features is given in table 2.

	Unigrams	Bigrams	Trigrams
Overall number of features	18257	89788	128699
Features after cutoff	3182	3099	1718
Features selected by the users	1391	1368	846

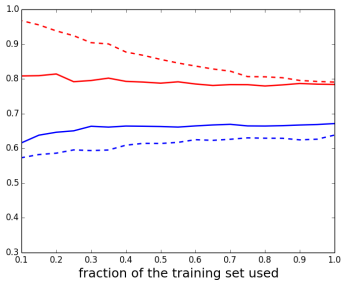
Table 2: Information about the features

We used a simple Naive Bayes classifier, with a small smoothing parameter. We also checked other classification algorithms- random forests, logistic regression, and multiclass SVM, (with $\|\cdot\|_1$-regularization and $\|\cdot\|_2$-regularization), but the results of the Naive Bayes predictor were the highest for both feature spaces.

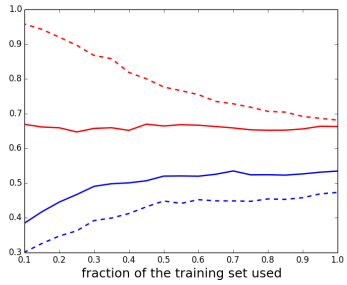
5.3 Results

The results that we will present are the results of the unigram model. The test scores of the other language models (unigram+bigrams and unigram+bigram+trigram) are almost identical for both feature spaces, and the training scores gets higher with the model complexity, as expected. Since our training set only contains approximately 8000 instances, we chose to present the results of the simplest model, so that the number of features would be comparable to the number of instances.

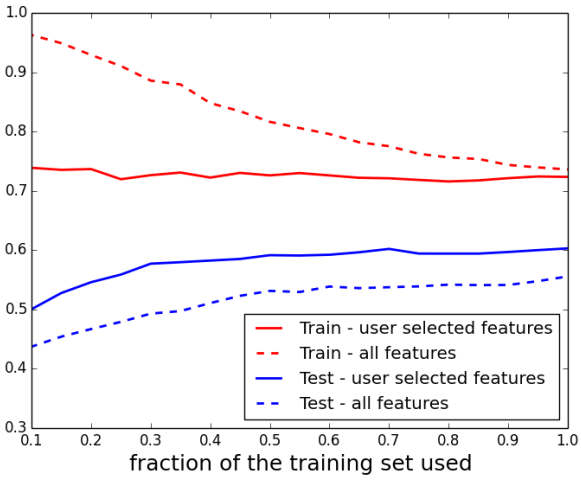
The results of both variants are presented in figure 2. As can be seen by the test scores, our algorithm outperforms the other variant which does not uses the additional information. The difference in test performance is approximately constant across different training sizes. Getting back to our claims - regarding the approximation error, by looking at the final training scores (using the larger training set possible), it can seen that both variants are



(a) F1-scores for positive samples



(b) F1-scores for negative samples



(c) Averaged F1-scores

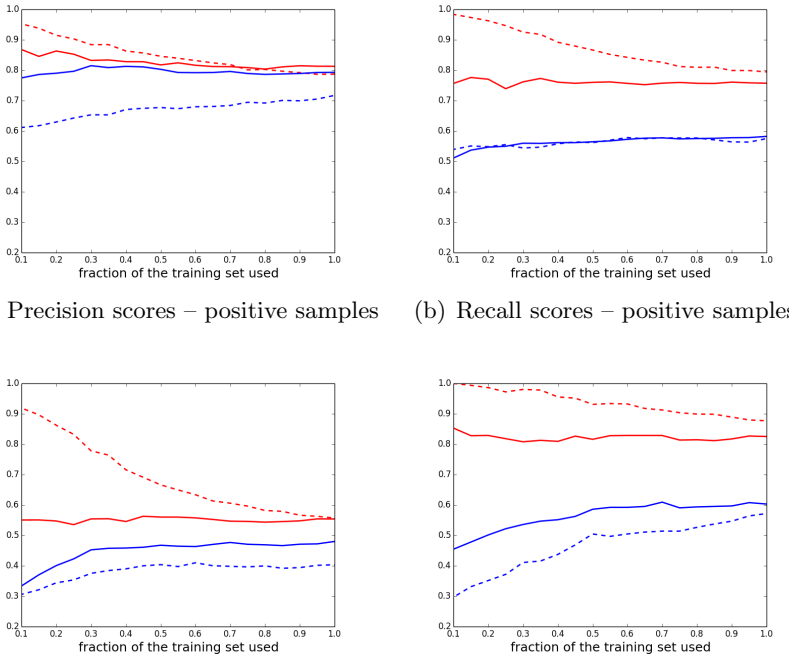
Figure 2: Train (red) and test (blue) $F1$-scores for a Naive Bayes classifier using the entire feature space (dashed lines) compared to using the queries-acquired feature space (continues lines): (a) for positive samples, (b) for negative samples and (c) average of positive and negative scores.

almost identical in all of the measurements. This fact indicates that we did not increase the approximation error. Regarding the improvement of estimation error, this can be seen clearly by looking at the gap between the test scores and the train scores. The gap in the query acquired model is smaller than the gap in the other model.

5.3.1 Precision and Recall

Additional interesting properties can be seen in the precision and recall graphs (figure 3). For example, by looking at the results for positive samples (a & b) we can see that the improvement in the results from using the query model is almost only due to the improvement in the precision scores. If we only use 10% of the data, the query model reaches 0.77 test precision, while the non-query model only reaches 0.71 test precision score

even when using the whole data set. Another interesting property that can be seen is that when a small training set is used, the difference in the test scores between the query and non-query methods is about twice as large as the difference when the largest possible training set is used.



(a) Precision scores – positive samples (b) Recall scores – positive samples

(c) Precision scores – negative samples (d) Recall scores – negative samples

Figure 3: Train (red) and test (blue) precision and recall scores for a Naive Bayes classifier using the entire feature space (dashed lines) compared to using the queries-acquired feature space (continues lines). Top – positive samples, bottom – negative samples, left – precision and right – recall.

5.3.2 Over-fitting

When using the naive bayes algorithm, we estimate $\mathbb{P}(f|c)$ for every feature f and every label c. This term measures how much the appearance of f contributes to the fact that c is the correct label [8] . Using those terms, we can sort the features by an order which conveys their informativeness. Since our features are words (or bigrams or trgrams), we can get some interesting insights by looking at the most informative features that each variant uses. If we only look at the top of the list (the top 20), the chosen features by both variants are almost identical. But, if we look a bit further we see how the algorithm which uses the entire feature space, chooses some significant features which clearly over-fit the training data. Some example are : "nick", "lloyd", and "justin" in the unigram model, "saturday

[8] by the naive assumption that all of the features are independent given the label, this information is actually the only information we use in order to build the classifier

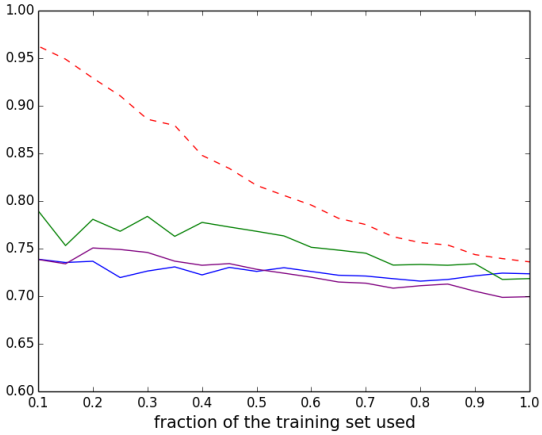
kitchen", "ghost rider", "ray lewis" in the bigram model and "rugby world cup" in the trigram model.

This over-fitting will obviously decrease as we increase the training size (and practically by checking the most informative features at different training sizes, the smaller the sample is, the more easy it is to find over-fitting features like the above). But as already stated, generally in Natural Language Processing it is much harder to acquire a large labeled data set. Therefore a method that avoids or significantly decrease this kind of over-fitting will be of high value.

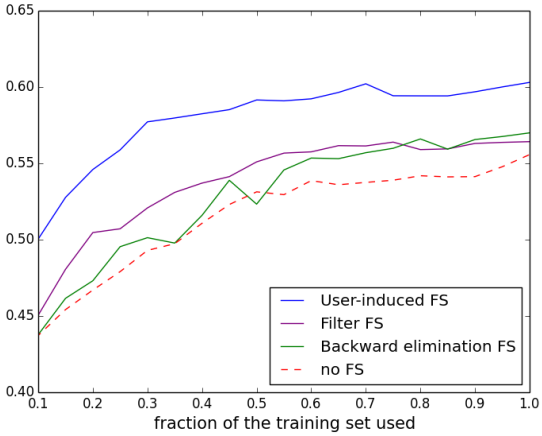
5.4 Comparing to other Feature Selection Methods

A question that can be raised is whether the improvement in the results is just an effect of the feature selection itself, or that the fact that the features were selected by a query process is the important part. In order to answer this, we compared our algorithm to using other *automatic* feature selection techniques. We checked two feature selection methods - filter and backward elimination. For each training set, the number of features that the method was instructed to select was the same as the number of features chosen by the users on that set. The results are presented in figure 4. The training scores of the automatic feature selection techniques are much lower than the training score of using the entire feature space (and much more similar to those of our method already for small training sets). This fact is reasonable, as we use a much smaller hypothesis class. If we look at the test scores it can be seen that using other feature selection techniques does improve the test score a little when compared to no feature selection at all, but still lies well under the score of our query acquired features method.

Another feature selection method that we compared our results to was using a SVM classifier with $\|\cdot\|_1$-regularization, which is known to induce sparsity. Here again, using our query acquired feature set outperforms in all of the measurements.



(a) Averaged F1 train scores



(b) Averaged F1 test scores

Figure 4: Train (top) and test (bottom) averaged F1-scores for our method (blue) compared to other automatic feature selection techniques – filter method (purple) and backward elimination (green) – as well as no feature selection (dashed red).

6 Conclusion and Future Work

We have presented both theoretical and empirical evidence that local-membership queries are useful and beneficial. In the theoretical setup we have shown that even 1-local queries are stronger than the vanilla PAC model in an arguably natural problem. In the empirical setup we have demonstrated that by getting additional information from the users, significantly better results can be achieved. Moreover, the data in the experiment was created using crowdsourcing, and by asking very simple questions. This shows that getting extra knowledge can be an easy task.

Today, the use of the MQ model in practice is almost non-existent. Even the more popular models of active learning, pool-based or stream-based, are fairly rare. E.g., in a recent survey of annotation projects for natural language processing tasks, only 20% of the respondents stated they had ever decided to use active learning (Tomanek and Olsson, 2009). It seems that there is plenty of room for incorporating more profound human knowledge to the field of machine learning, especially since today this knowledge can be collected quite easily.

More concrete directions for future work include: developing, implementing and analyzing more algorithms that use (local) membership queries and investigating the strength and limitations of the general $O(1)$-local queries model. Some examples of open questions: Is the use of 2-local queries stronger than the use of 1-local queries on a natural environment? What are the limitations of a model that uses $O(1)$-local queries with comparison to the model of (Awasthi et al., 2012) that uses $\log(n)$-local queries?

References

Angluin, D. (1987). Learning regular sets from queries and counterexamples. *Information and computation*, 75(2):87–106.

Angluin, D. and Kharitonov, M. (1995). When won t membership queries help? *Journal of Computer and System Sciences*, 50(2):336–355.

Angluin, D., Krikis, M., Sloan, R. H., and Turán, G. (1997). Malicious omissions and errors in answers to membership queries. *Machine Learning*, 28(2-3):211–255.

Angluin, D. and Slonim, D. K. (1994). Randomly fallible teachers: Learning monotone dnf with an incomplete membership oracle. *Machine Learning*, 14(1):7–26.

Awasthi, P., Feldman, V., and Kanade, V. (2012). Learning using local membership queries. *arXiv preprint arXiv:1211.0996*.

Balcan, M.-F., Beygelzimer, A., and Langford, J. (2006). Agnostic active learning. In *Proceedings of the 23rd international conference on Machine learning*, pages 65–72. ACM.

Barbosa, L. and Feng, J. (2010). Robust sentiment detection on twitter from biased and noisy data. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 36–44. Association for Computational Linguistics.

Baum, E. B. (1991). Neural net algorithms that learn in polynomial time from examples and queries. *Neural Networks, IEEE Transactions on*, 2(1):5–19.

Baum, E. B. and Lang, K. (1992). Query learning can work poorly when a human oracle is used. In *International Joint Conference on Neural Networks*, volume 8.

Bisht, L., Bshouty, N. H., and Khoury, L. (2008). Learning with errors in answers to membership queries. *Journal of Computer and System Sciences*, 74(1):2–15.

Blum, A., Chalasani, P., Goldman, S. A., and Slonim, D. K. (1995). Learning with unreliable boundary queries. In *Proceedings of the eighth annual conference on Computational learning theory*, pages 98–107. ACM.

Blum, A. and Rudich, S. (1992). Fast learning of k-term dnf formulas with queries. In *Proceedings of the twenty-fourth annual ACM symposium on Theory of computing*, pages 382–389. ACM.

Bshouty, N. H. (1995). Exact learning boolean functions via the monotone theory. *Information and Computation*, 123(1):146–153.

Daniely, A., Linial, N., and Shalev-Shwartz, S. (2013). More data speeds up training time in learning halfspaces over sparse vectors. In *Advances in Neural Information Processing Systems*, pages 145–153.

Daniely, A., Linial, N., and Shalev-Shwartz, S. (2014). From average case complexity to improper learning complexity. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, pages 441–448. ACM.

Daniely, A. and Shalev-Shwatz, S. (2014). Complexity theoretic limitations on learning dnf's. *arXiv preprint arXiv:1404.3378*.

Dasgupta, S. (2004). Analysis of a greedy active learning strategy. In *Advances in neural information processing systems*, pages 337–344.

Davidov, D., Tsur, O., and Rappoport, A. (2010). Enhanced sentiment learning using twitter hashtags and smileys. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 241–249. Association for Computational Linguistics.

Druck, G., Settles, B., and McCallum, A. (2009). Active learning by labeling features. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pages 81–90. Association for Computational Linguistics.

Ehrenfeucht, A. and Haussler, D. (1989). Learning decision trees from random examples. *Information and Computation*, 82(3):231–246.

Feldman, V. (2009). On the power of membership queries in agnostic learning. *The Journal of Machine Learning Research*, 10:163–182.

Freund, Y. (1995). Boosting a weak learning algorithm by majority. *Information and computation*, 121(2):256–285.

Hu, M. and Liu, B. (2004). Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM.

Jackson, J. (1994). An efficient membership-query algorithm for learning dnf with respect to the uniform distribution. In *Foundations of Computer Science, 1994 Proceedings., 35th Annual Symposium on*, pages 42–53. IEEE.

Kearns, M. and Valiant, L. (1994). Cryptographic limitations on learning boolean formulae and finite automata. *Journal of the ACM (JACM)*, 41(1):67–95.

Kim, S.-M. and Hovy, E. (2004). Determining the sentiment of opinions. In *Proceedings of the 20th international conference on Computational Linguistics*, page 1367. Association for Computational Linguistics.

Klivans, A. R., Sherstov, A., et al. (2006). Cryptographic hardness for learning intersections of halfspaces. In *Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on*, pages 553–562. IEEE.

Kouloumpis, E., Wilson, T., and Moore, J. (2011). Twitter sentiment analysis: The good the bad and the omg! *Icwsm*, 11:538–541.

Kushilevitz, E. and Mansour, Y. (1993). Learning decision trees using the fourier spectrum. *SIAM Journal on Computing*, 22(6):1331–1348.

Nakov, P., Kozareva, Z., Ritter, A., Rosenthal, S., Stoyanov, V., and Wilson, T. (2013). Semeval-2013 task 2: Sentiment analysis in twitter.

Pak, A. and Paroubek, P. (2010). Twitter as a corpus for sentiment analysis and opinion mining. In *LREC*, volume 10, pages 1320–1326.

Pang, B. and Lee, L. (2004). A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd annual meeting on Association for Computational Linguistics*, page 271. Association for Computational Linguistics.

Pang, B. and Lee, L. (2008). Opinion mining and sentiment analysis. *Foundations and trends in information retrieval*, 2(1-2):1–135.

Raghavan, H. and Allan, J. (2007). An interactive algorithm for asking and incorporating feature feedback into support vector machines. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 79–86. ACM.

Raghavan, H., Madani, O., and Jones, R. (2005). Interactive feature selection. In *IJCAI*, volume 5, pages 841–846.

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

Settles, B. (2011). Closing the loop: Fast, interactive semi-supervised annotation with queries on features and instances. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1467–1478. Association for Computational Linguistics.

Sloan, R. H. and Turán, G. (1994). Learning with queries but incomplete information. In *Proceedings of the seventh annual conference on Computational learning theory*, pages 237–245. ACM.

Tomanek, K. and Olsson, F. (2009). A web survey on the use of active learning to support annotation of text data. In *Proceedings of the NAACL HLT 2009 Workshop on Active Learning for Natural Language Processing*, pages 45–48. Association for Computational Linguistics.

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