# Learning Switching Concepts

**Avrim Blum**
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
avrim@theory.cs.cmu.edu

**Prasad Chalasani**
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
chal@cs.cmu.edu

## Abstract

We consider learning in situations where the function used to classify examples may switch back and forth between a small number of different concepts during the course of learning. We examine several models for such situations: oblivious models in which switches are made independent of the selection of examples, and more adversarial models in which a single adversary controls both the concept switches and example selection.

We show relationships between the more benign models and the $p$-concepts of Kearns and Schapire, and present polynomial-time algorithms for learning switches between two $k$-DNF formulas. For the most adversarial model, we present a model of success patterned after the popular competitive analysis used in studying on-line algorithms. We describe a randomized query algorithm for such adversarial switches between two monotone disjunctions that is "1-competitive" in that the total number of mistakes plus queries is with high probability bounded by the number of switches plus some fixed polynomial in $n$ (the number of variables).

We also use notions described here to provide sufficient conditions under which learning a $p$-concept class "with a decision rule" implies being able to learn the class "with a model of probability."

## 1 INTRODUCTION

The standard problem considered in machine learning theory is that of learning a fixed concept from some class $\mathcal{C}$. In this paper we examine a generalization in which the concept may switch between a small number of different concepts during the course of learning. For example, imagine learning what type of food a person likes where examples are different foods, and classification is positive if the food is eaten. In such a case, the learner's observations will likely be different depending on whether the person is hungry or not hungry, or whether it is morning or evening. Instead of having a fixed concept, a better description of the situation may be that there are two concepts $c_1$ and $c_2$, and classification occasionally switches between one and the other.

This type of situation is similar to that considered by Helmbold and Long [HL91] in which a concept may drift over time, but there are several important differences between our focus and theirs. We restrict the "active concept" to switch between only a small number of concepts instead of drifting through the entire class, but we allow the switches to occur more rapidly and drastically. For example, in our case the concept might switch on average every third example, so each "run" of examples is much less that the sample size needed to learn a concept; just looking at data more recent than some cutoff point need not produce a large set consistent with any individual target. Our main goal is to produce polynomial-time algorithms for simple classes, though we will also discuss somewhat the use of a minimum disagreements oracle.

Experimental work on learning interleaved functions has been done by E. Levin [Lev91]. Ar et al. [ALRS] have examined a similar problem of identifying a set of polynomials over a finite field where each point is assigned a value by one of the polynomials.

We consider two main models for how concept switches are made. The more benign one is the *oblivious adversary* model. Here, examples are selected from a fixed distribution and an adversary decides when to switch the active concept. However, the adversary must do so *in advance*, choosing the entire path of the active concept before any selections from the distribution have been made. In this paper, we will in fact suppose the target is switching between only two concepts. We say that we *learn* a pair of target concepts if we find $\epsilon$-close hypotheses for each one. For this model, we


COLT'92-7/92/PA,USA
© 1992 ACM 0-89791-498-8/92/0007/0231...$1.50

give (section 6) a polynomial-time algorithm to learn disjunctions[1] so long as with high probability (1) both concepts are represented a poly-fraction of the time on any sufficiently large sample, and (2) the average "run length" (the number of examples in a row classified by the same concept) is at least $2+\alpha$ for some $\alpha > 0$.

We also consider, in section 3, a more malicious model we call the *strong adversary* model, in which an adversary controls both the selection of examples and the switching between concepts, and may do so based on the entire past conversation with the learner. Our style of analysis of this model is motivated by "competitiveness" models of on-line algorithms (e.g. [BLS87], [MMS90], [BDBK+90]). We imagine that the adversary pays a cost of 1 for each switch, and present a "1-competitive" randomized algorithm that uses *membership queries* for the class of monotone disjunctions. Specifically, with high probability the total number of mistakes plus queries made by the algorithm is at most $p(n) + s$ where $p$ is a fixed polynomial and $s$ is the number adversary switches. Note that without queries, learning in this model contains as a special case the problem of learning disjunctions with worst-case false-negative errors (one of the concepts could be "false").

In addition to the above problems, we also consider (section 4) the related problem of learning when each example is *independently* classified according to one of a small number of concepts, according to probabilities that favor one of them. For example, there might be two concepts $c_1$ and $c_2$, and each example has a 60% chance of being classified by $c_1$ and a 40% chance of being classified by $c_2$. This situation might occur if there is a relevant variable that is hidden from the learner, and its value is independent of the visible variables and biased in one direction (see [KS90], [KSS92]). We call such a situation a *mixture* of two concepts, and our goal is to approximate one or both of them. One of our motivations for studying mixtures is that we use algorithms for learning in this model in a critical way in our algorithms for the switching concept models.

The mixture model is similar to the Angluin and Laird [AL88] noise model, except that here the "noise" is systematic in that it is consistent with some other concept in the class. It could also be thought of as a weaker version of Sloan's "malicious misclassification" model [Slo88] and actually is a special case of Kearns and Schapire's $p$-concepts [KS90], though the notions of success are somewhat different. In fact, we show how Kearns and Schapire's result on learning probabilistic decision lists of decreasing probabilities can be applied to learn mixtures of $k$ disjunctions with a "model of probability," and actually retrieve the concepts when $k = 2$ so long as the two probabilities differ by at least some $\alpha > 0$. We also use notions of mixtures to provide sufficient conditions where learning a $p$-concept class with a model of probability is as easy as learning the class with a decision rule (Section 7).

[1] All the results given for learning disjunctions without queries hold for $k$-CNF and $k$-DNF formulas as well, by standard transformations.

## 2 NOTATION AND DEFINITIONS

Let $V$ be a set of $n$ variables $\{x_1, \ldots, x_n\}$. An example $x$ is an assignment of 0 or 1 to each $x_i \in V$, and let $X$ be the set of all examples. We will often just write $x_i = 1$ or $x_i = 0$ to mean $x(x_i) = 1$ or $x(x_i) = 0$ when the example is clear from context. A labeled example is a pair $(x, l)$ where $x \in X$ and $l \in \{0, 1\}$. A concept is a boolean function over examples and a concept class is a collection of concepts. For a disjunction $c$, define $R(c)$ to be the set of all variables disjoined in $c$. So, $R(c)$ is the set of *relevant* variables.

A $p$-concept $\mathbf{c}$ (defined by Kearns and Schapire [KS90]) is a function over $X$ with values in $[0, 1]$, with the value of $\mathbf{c}(x)$ interpreted as a probability. An example $x$ classified by $\mathbf{c}$ is given label 1 with probability $\mathbf{c}(x)$ and 0 with probability $1 - \mathbf{c}(x)$. We will denote $p$-concepts by bold letters, to distinguish them from boolean-valued functions. A $p$-concept $\mathbf{h}$ is an "$\epsilon$-good model of probability" of $\mathbf{c}$ with respect to a distribution $\mathcal{D}$ if $\mathbf{Pr}_{x \in \mathcal{D}}[|\mathbf{h}(x) - \mathbf{c}(x)| \leq \epsilon] \geq 1 - \epsilon$. Thus, the value of $\mathbf{h}$ must be near that of $\mathbf{c}$ on most points $x$. An algorithm learns a $p$-concept class "with a model of probability" if for any $p$-concept $\mathbf{c}$ in the class and any distribution $\mathcal{D}$, given access to examples from $\mathcal{D}$ classified by $\mathbf{c}$, with high probability it finds such an $\mathbf{h}$. Note that the learner only sees the boolean-valued classification of an example $x$, and not the real value $\mathbf{c}(x)$.

A *mixture* $\mathrm{MIX}(c_1, \ldots, c_k; p_1, \ldots, p_k)$ where $c_1, \ldots, c_k$ are concepts and $p_1, \ldots, p_k$ are probabilities that sum to 1, is the following oracle. Given example $x$, one of the concepts $c_i$ is chosen according to the probabilities $p_i$, and then $c_i(x)$ is returned as the classification. Note that for each example $x$, the probability that $x$ is classified positive is simply $\sum_i c_i(x)p_i$. Thus, a mixture oracle is a type of $p$-concept.

For simplicity, for $k = 2$, we use the shorthand $\mathrm{MIX}(c_1, c_2, \nu)$ to denote $\mathrm{MIX}(c_1, c_2; 1-\nu, \nu)$ (we do it this way since we are thinking of $\nu$ as less than $1/2$ and $c_1$ as the "main" concept). We define the class $\mathrm{MIX}(\mathcal{C}, \nu_b)$ for $\nu_b \leq \frac{1}{2}$ to be the set of all mixtures $\mathrm{MIX}(c_1, c_2, \nu)$ such that $c_1, c_2 \in \mathcal{C}$ and $\nu \in [\frac{1}{2} - \nu_b, \nu_b]$. We say that we PAC-learn the class $\mathrm{MIX}(\mathcal{C}, \nu_b)$ if given access to a MIX oracle in the class and a distribution $\mathcal{D}$ over $X$, and given $\epsilon, \delta > 0$, with probability at least $1 - \delta$ we can find $\epsilon$-close hypotheses to both concepts. We will give algorithms that run in time polynomial in $n$, $1/\epsilon$, $1/\delta$, and $1/(\frac{1}{2} - \nu_b)$. For mixtures of $k > 2$ concepts, we will only look at learning the "majority" concept (i.e., the one, if any, with $p_i > 1/2$).

A *switching concept oracle* is like a mixture, except we introduce a time-dependency. In the oblivious adversary switching model, an adversary specifies in advance, *before examples are drawn* from the distribution $\mathcal{D}$, a se-

quence $\{c_i\}$ of target concepts. Then, a sequence $\{x^i\}$ of examples is drawn from $\mathcal{D}$ and each example $x^i$ is classified by the corresponding target concept $c_i$. In our algorithms we will only examine switches between two concepts, and will require that the adversary satisfy the following properties. (1) We are given a lower bound $\beta_l > 0$ such that in any sufficiently large sequence of examples, at least a $\beta_l$ fraction are classified by each concept, and (2) we are given a $\beta_u < 1/2$ such that in any sufficiently large sample, the average run length of any concept is at least $1/\beta_u$. The first property ensures that a sufficient number of examples of of each concept are seen, and the second ensures that the adversary is not switching at almost every other example. This model allows considerable leeway in specifying switching patterns.

In the strong adversary model, the adversary need not be oblivious, and in addition it both selects when to switch the concepts and chooses the examples. There is no distribution. Learning in this model proceeds in rounds much like the standard on-line learning models. In each round, first the adversary decides whether to switch the active concept. Then, the learner asks to receive an unlabeled example or makes a membership query if queries are allowed. In the former case, the learner then makes a prediction and is told if it was correct; in the latter, the learner is given the classification of the query. We then begin a new round. Note that in this model, the adversary may decide to switch based on the interaction so far, but cannot switch, for instance, between receiving a membership query and responding to it. Our goal in this model is to bound the number of mistakes plus queries made by the learner based on the number of switches of the adversary.

# 3 LEARNING SWITCHING CONCEPTS WITH QUERIES

In this section, we consider learning under the strong adversary switching model, but to compensate, allow the learner to make membership queries. As noted earlier, without queries this problem is at least as hard as the problem of learning disjunctions with worst-case false-negative errors, for which some evidence of intrinsic difficulty has been given by Kearns and Li [KL88].

Our analysis is patterned after the popular "competitive analysis" model of online algorithms (e.g. [BLS87], [MMS90], [BDBK+90]). We will charge the adversary a cost of 1 for each switch, and we will charge the algorithm for the total number of mistakes plus queries made. Note that if there are two concepts, an algorithm that knows both exactly need only make 1 mistake per switch. It just predicts according to its current concept until a mistake is made and then switches to the other; we call this the "standard steady-state" algorithm. In analogy to the on-line algorithms literature, we will say an algorithm is $t$-competitive on a sequence if the number of mistakes plus queries is bounded by $p(n) + ts$, where $p$ is a polynomial and $s$ is the number of adversary switches. We say a randomized algorithm is $t$-competitive for an adversarially-switched class if it is $t$-competitive for any pair of concepts chosen, with high probability over the random choices of the algorithm. We present here a 1-competitive randomized algorithm for learning a pair of monotone disjunctions in the strong-adversary switching model. Specifically, given $\delta > 0$, with probability at least $1 - \delta$ the number of mistakes plus queries of the algorithm is $O(n^2 \log(n/\delta)) + s$. The algorithm's running time per round is polynomial in $n$ and $\log(1/\delta)$.

We begin with a high-level description of the algorithm. In its first stage, the algorithm makes a large number of queries, selected randomly according to a specific distribution. With high probability it learns one concept exactly and learns the other exactly as well if it has been used for classification a fair fraction of the time. Of course, it is possible that one concept has been used only very infrequently for classification, so one cannot hope to always learn both.

In the second stage, the algorithm predicts according to the first concept $c_1$ learned, and whenever its prediction is wrong, makes a query, hoping to gain information about the second concept $c_2$. We show that with high probability, after making sufficiently (polynomially) many queries to $c_2$, the algorithm will both recognize this fact—so it can stop making the queries—and exactly learn $c_2$. We also show that except for mistakes and queries associated with the queries to $c_2$, the algorithm makes at most one mistake and one query for every *two* switches of the adversary, which makes it 1-competitive.

For convenience, let $\langle x_i \rangle$ denote the example that sets only $x_i$ to 1, and similarly define $\langle x_i, x_j \rangle$ as the example that sets only $x_i$ and $x_j$ to 1. The first stage of the algorithm is as follows.

**Algorithm** Query-Learn, Stage One:

*Given:* $\delta > 0$.

1. *Make* $m = kn^2 \log(n/\delta)$ *queries (*$k$ *is a sufficiently large constant) each selected independently according to the following distribution:*

   *Each example* $\langle x_i \rangle$ *is selected with probability* $\frac{1}{2n}$. *Each example* $\langle x_i, x_j \rangle$ $(i \neq j)$ *is selected with probability* $\frac{1}{n(n-1)}$.

2. *For each* $i$ *calculate* $f^+(x_i) =$ (*number of queries* $\langle x_i \rangle$ *that were classified as 1*)/$(\frac{m}{2n})$.

3. *If no index* $i$ *has* $f^+(x_i) \in [\frac{1}{4}, \frac{3}{4}]$, *then let* $h$ *be the disjunction of all* $x_i$ *such that* $f^+(x_i) > 1/2$. *Output* $h$.

4. *Otherwise, pick* $i$ *such that* $f^+(x_i) \in [\frac{1}{4}, \frac{3}{4}]$.

   *Let* $h$ *be the disjunction of all* $x_j$ *such that query* $\langle x_i, x_j \rangle$ *was never classified as negative.*

*Let $h'$ be the disjunction of all $x_j$ such that either all queries $\langle x_j \rangle$ were classified as positive, or else $f^+(x_j) > 0$ and $x_i \notin R(h)$.*

*Output both $h$ and $h'$.*

**Lemma 1** *For any pair of strong-adversary switched monotone disjunctions and any $\delta > 0$, with probability $1 - \delta$ Algorithm* Query-Learn, Stage One *will output either one or both of the target concepts.*

**Proof:** Algorithm Query-Learn, Stage One makes $m$ queries. Since each query is selected independently from a fixed distribution, the only relevant choice of the adversary is the number $m_1$ of queries for which $c_1$ is used for classification. This can be seen by thinking of the algorithm as having two random tapes, one used when the target is $c_1$ and the other when the target is $c_2$. Let us first suppose that the adversary chooses $m_1$ *in advance*, and without loss of generality let us say $m_1/m \geq 1/2$.

For each $x_i \in R(c_1)$, the expected value of $f^+(x_i)$ is at least $m_1/m \geq 1/2$. (The expected value is 1 if $x_i \in R(c_2)$ as well.) Similarly, for each $x_i \notin R(c_1)$, the expected value of $f^+(x_i)$ is at most $1 - m_1/m \leq 1/2$. Chernoff bounds imply that with probability $1 - ne^{-O(m/n)} > 1 - \delta/4$, no $f^+(x_i)$ for $x_i \in R(c_1)$ will be less than $1/4$, and no $f^+(x_i)$ for $x_i \notin R(c_1)$ will be greater than $3/4$. Thus, with high probability, if the algorithm exits in Step 3 then $h = c_1$. Notice also that if $m_1 > 7m/8$, then with high probability (another $1 - \delta/4$) each $x_i \in R(c_1)$ has $f^+(x_i) > 3/4$, and all others have $f^+(x_i) < 1/4$, so the algorithm *will* exit in Step 3. Thus, we only need to analyze Step 4 for $m_1 \leq 7m/8$.

If the algorithm continues to Step 4, then with high probability the variable $x_i$ chosen such that $f^+(x_i) \in [1/4, 3/4]$ is relevant to exactly one of the two concepts (if it was relevant to both, then with high probability $f^+(x_i)$ is very near 1). Let's say that $x_i$ is relevant to $c_1$. In that case, for each $x_j \in R(c_2)$, the query $\langle x_i, x_j \rangle$ will never be answered negative. So, hypothesis $h$ will contain all variables in $R(c_2)$. In addition, if $x_j \notin R(c_2)$, then the query $\langle x_i, x_j \rangle$ *will* be answered negative if it is asked when classification is according to $c_2$. For $m_1 \leq 7m/8$, this occurs with probability at least $1 - [1 - \frac{1}{n(n-1)}]^{m/8}$ which is at least $1 - \delta/4n$ for sufficiently large $k$. So, with high probability $h = c_2$. Given that this occurs, with high probability $h' = c_1$. The same analysis holds if $x_i$ was relevant to $c_2$.

So far we have been assuming that $m_1$ was chosen in advance, but we can make all failure rates sufficiently small ($\delta/m$) that for all values of $m_1$ simultaneously, the chance of failure is at most $\delta$. That is, if we think of the algorithm as having two random tapes as described above, then even if the adversary determines $m_1$ after seeing the entire contents of the tapes, it still can only foil the procedure with probability $\delta$. ■

The second stage of the algorithm is as follows.

**Algorithm** Query-Learn, Stage Two:

*Given: concept $c_1$, $\delta > 0$. (If we are given both concepts, we just run the standard steady-state algorithm).*

1. $S \leftarrow 0$
2. *Repeat until $S = kn \log(n/\delta)$:*
   (a) *Predict according to $c_1$ until a mistake is made.*
   (b) *Make a query $\langle x_i \rangle$, with $x_i$ chosen uniformly at random from $\{x_1, x_2, ...x_n\}$.*
   (c) *If the answer to the query is not the same as $c_1(\langle x_i \rangle)$, then $S \leftarrow S + 1$.*
3. *Let $h_2$ be the disjunction of the following variables:*
   - *Variables $x_i \notin R(c_1)$ such that query $\langle x_i \rangle$ was classified as positive at least once, and*
   - *Variables $x_i \in R(c_1)$ such that query $\langle x_i \rangle$ was* **always** *classified as positive.*
4. *Run the standard steady-state algorithm on $c_1$ and $h_2$.*

**Lemma 2** *For any pair of strong-adversary switched monotone disjunctions $(c_1, c_2)$ such that $c_1$ is given to the learner, with probability $1 - \delta$ Algorithm* Query-Learn, Stage Two *will make at most $O(n^2 \log(n/\delta)) + s$ mistakes plus queries where $s$ is the number of switches of the adversary.*

**Proof:** If $c_1 = c_2$, the algorithm stays in step 2(a) and the claim trivially holds. So, assume $c_1 \neq c_2$, so that there is at least one variable in the symmetric difference $R(c_1) \Delta R(c_2)$. For $i = 1, 2$, we say that a query is a $c_i$-query when the adversary uses concept $c_i$ to classify it. Also, a $c_2$-query $x$ is "informative" when the answer is different from $c_1(x)$. The counter $S$ in step 2 counts precisely the number of informative $c_2$-queries made. Since $|R(c_1) \Delta R(c_2)| \geq 1$, if a $c_2$-query is made in step 2(b), there is at least a $1/n$ chance that it is informative. So, if we consider a sequence of $2kn^2 \log(n/\delta)$ $c_2$-queries selected according to the distribution of step 2(b), with high probability both (A) at least $kn \log(n/\delta)$ are informative, and (B) within the first $kn \log(n/\delta)$ $c_2$-queries, informative queries $\langle x_i \rangle$ have been made at least once for each $x_i \in R(c_1) \Delta R(c_2)$. (The probability of failure is $O(n \cdot (\delta/n)^{O(k)})$.) This implies that even if the adversary could see in advance the entire list of random choices the algorithm would make when performing $c_2$-queries, with high probability either the algorithm proceeds to step 3 and by (B) the hypothesis $h_2$ created in step 3 exactly equals $c_2$, or else the adversary does not allow the algorithm to go to step 3 by not letting it make $2kn^2 \log(n/\delta)$ $c_2$-queries.[2]

[2]The reason for the care here is a subtlety, which can be illustrated by the following game: imagine performing a random walk on a line, but where an adversary can stop you at any time $t \leq 100$. One might like to say: "if the adversary allows you to make 100 steps, then with high probability you will have made about 50 steps to the right." However, this

Thus, it remains only to show that the number of mistakes plus queries made in step 2 is bounded by the number of adversary switches plus a constant times the number of $c_2$-queries. To see this, consider a sequence $\langle m_1, q_1, m_2, q_2, ...\rangle$ of mistakes and queries. Each mistake $m_i$ is made while the classification is according to $c_2$. So, if $q_i$ is a $c_1$-query then the adversary must have switched to $c_1$ for $q_i$ and then back to $c_2$ before $m_{i+1}$, and we can charge both $q_i$ and $m_{i+1}$ to the switches. If $q_i$ is a $c_2$-query, then we charge $q_i$ and $m_{i+1}$ to the query, plus we perhaps need to pay for mistake $m_1$. So, the number not "paid for" by adversary switches is at most 1 plus twice the number of $c_2$-queries. Since with high probability we make only $O(n^2 \log(n/\delta)$ $c_2$-queries, this achieves the bounds claimed. ■

The above two lemmas prove the following theorem.

**Theorem 3** *Algorithm* Query-Learn, Stage One, *followed by* Query-Learn, Stage Two *is 1-competitive for the class of strong-adversary switched disjunctions.*

# 4 LEARNING CONCEPTS FROM A MIXTURE

In the mixture model MIX$(c_1, c_2, \nu)$, each example selected from $\mathcal{D}$ is classified by $c_1$ with probability $1-\nu$ and by $c_2$ with probability $\nu$. This is similar to Angluin and Laird's noise model [AL88] in which there is a single target concept, but each example has fixed probability $\nu$ of being classified by its complement. It is also a special case of Sloan's malicious misclassification (MMC) model [Slo88] in which with probability $\nu$, an adversary may decide the example's classification.

Angluin and Laird describe an algorithm to learn the class of monotone disjunctions in their noise model that proceeds essentially as follows.[3] Take a large sample of data, and for each variable $x_i$ that is seen set to 1 reasonably often, calculate the fraction of examples with $x_i = 1$ that are positive. Then produce as hypothesis a disjunction of all those whose value is near the maximum. This algorithm may fail, however, when applied to 2-mixtures. For example, consider $c_1 = x_1 \vee \ldots \vee x_{n/2}$, $c_2 = x_{n/2+1} \vee \ldots \vee x_n$, $\nu = 0.2$ and a distribution $\mathcal{D}$ as follows. With probability 0.1 the example sets to 1 only a random $x_i \in R(c_1)$, with probability 0.8 the example sets to 1 only a random $x_j \in R(c_2)$, and with the remaining 0.1 probability, the example sets to 1 one random $x_i \in R(c_1)$ and 100 random $x_j \in R(c_2)$. One can calculate that for each $x_i \in R(c_1)$ the probability an example with $x_i = 1$ is positive is 0.9, but for each $x_j \in R(c_2)$, the probability is about 0.94. So, taking the OR of the variables of highest value does not produce a good hypothesis (if you let $h$ be the disjunction of all $x_i$, then the probability $h$ labels an example correctly is just 0.34, and if you take $h$ to be the disjunction of the $x_j \in R(c_2)$ the probability is lower).

Instead, our approach is the following. Consider a pair of variables $x_i \in R(c_1)$ and $x_j \in R(c_2)$. For such a pair, every example seen satisfying $x_i x_j$ will necessarily be positive. So, if we first create a 2-DNF $h_{12}$ of all pairs $x_i x_j$ such that no example setting both to 1 has appeared negative, then filtering $\mathcal{D}$ through $h_{12}(x) = 0$ will yield a distribution under which no example satisfies both $c_1$ and $c_2$. Under this filtered distribution, the Angluin-Laird style analysis *can* be applied, and the concepts learned can then be OR'ed with $h_{12}$. In fact, a generalization of this procedure is used by Kearns and Schapire [KS90] to learn the class of "$p$-decision lists with decreasing probabilities," and it turns out we can directly use their theorems to get the results we need.

## 4.1 MIXTURES AS PROBABILISTIC DECISION LISTS

We show here that mixtures of disjunctions can be written as a special kind of probabilistic concept called a "probabilistic decision list ($p$-DL) with decreasing probabilities," introduced by Kearns and Schapire in [KS90].

$p$-DL's are probabilistic analogs of standard decision lists [Riv87]. A 1-p-DL c is given by a list $(x_{i_1}, r_1), ..., (x_{i_n}, r_n)$ where the $x_{i_j}$ are distinct variables, and each $r_i \in [0,1]$. For any example $x$, $\mathbf{c}(x)$ is defined to be $r_j$ where $j$ is the least index such that $x_{i_j} = 1$. In other words, the $n$ variables are tested one by one in the order specified by the list, until a variable $x_{i_j}$ is found which is 1 in the example $x$. The corresponding real number $r_j$ is the probability that $x$ is positive. A $k$-p-DL is defined in the same way, except that it has entries of the form $(t, r)$ where $t$ (called a term) can be a product of up to $k$ variables. That is, $\mathbf{c}(x)$ is the $r$-value of the first entry whose term $t$ is satisfied by $x$.

A *$k$-p-DL with decreasing probabilities* is one with $r_1 \geq \ldots \geq r_s$, where $s$ is the number of entries in the $k$-p-DL.

To see how we might view a mixture of monotone disjunctions as a p-DL with decreasing probabilities, consider the 2-mixture MIX$(c_1, c_2, \nu)$, for $\nu < 1/2$. The probability $\mathbf{c}(x)$ that an example $x$ is labeled 1 by MIX is given by the following rules:

1. If $x$ satisfies $(c_1 \wedge c_2)$, then $\mathbf{c}(x) = 1$.
2. Otherwise, if $x$ satisfies $c_1$ then $\mathbf{c}(x) = 1 - \nu$.
3. Otherwise, if $x$ satisfies $c_2$ then $\mathbf{c}(x) = \nu$.
4. Otherwise, $\mathbf{c}(x) = 0$.

is false because the adversary's strategy might be to stop the game after your first step to the left (so if you do make 100 steps, at least 99 were to the right). What is true, which can be seen by considering the more powerful adversary that can see your entire sequence of 100 coin tosses in advance, is that with high probability either the adversary stops you before $t = 100$ or by $t = 100$ you will have made about 50 steps to the right.

[3] In Sloan [Slo88] it is claimed that Angluin and Laird's algorithm can be applied directly to the MMC model. However, Sloan [Slo] has recently discovered a bug in the transformation, and currently there is no known poly-time algorithm for learning disjunctions in the MMC model.

These rules translate to a 2-p-DL in a natural way. Rule 1 gives rise to entries $(t, 1)$ for each term $t$ in the 2-DNF expansion of $(c_1 \wedge c_2)$. Rule 2 gives rise to entries $(t, 1-\nu)$, for each $t \in R(c_1)$, and similarly for rule 3. Finally rule 4 gives rise to the entry $(1, 0)$.

One can convert backwards from this 2-p-DL to the disjunctions $c_1$ and $c_2$, if $\nu < 1/2$. We have $c_1(x) = 1$ iff $\mathbf{c}(x) = 1$ or $\mathbf{c}(x) = 1-\nu$. Also, $c_2(x) = 1$ iff $\mathbf{c}(x) = 1$ or $\mathbf{c}(x) = \nu$. Kearns and Schapire define the *projection* of a $p$-concept $\mathbf{c}$ as a boolean-valued function $\pi_{\mathbf{c}}$ which is 1 exactly when $\mathbf{c}(x) \geq 1/2$. Thus, for $\nu < 1/2$, $c_1$ is simply the projection of the $p$-concept $\mathbf{c}$ corresponding to the mixture.

In fact any mixture of $k$ monotone disjunctions can be written as a $k$−p-DL with decreasing probabilities.

**Theorem 4** *A mixture of $k$ monotone disjunctions can be written as a $k$-p-DL with decreasing probabilities.*

**Proof:** Consider a mixture of the monotone disjunctions $c_1, c_2, \ldots, c_k$, with corresponding probabilities $p_1, p_2, \ldots, p_k$ all greater than 0. Construct a $k$-p-DL as follows. Let $\mathcal{S} = \{S_1, S_2, \ldots, S_{2^k-1}\}$ be the collection of all non-empty subsets of the index set $\{1, 2, \ldots, k\}$. For $i = 1, 2, \ldots, 2^k - 1$, define $P_i = \sum_{j \in S_i}(p_j)$, and $T_i$ = the set of terms in the DNF expansion of the CNF $\bigwedge_{j \in S_i}(c_j)$. For each $i$ create a list $L_i$ consisting of an entry $(t, P_i)$ for every $t \in T_i$. Merge all the lists $L_i$ into a single list, and sort it in order of decreasing $P_i$. Finally, add the entry $(1, 0)$ at the end of the list. This is the final $k$-p-DL.

By definition, this is a p-DL with decreasing probabilities. We must show that it models the $k$-mixture. Suppose that the first entry satisfied by an example $x$ is $(t, P_i)$ for some $i$. Thus, $x$ satisfies the conjunction of $c_j$ for $j \in S_i$; also for any set $S_j$ that strictly contains $S_i$, the probability-sum $P_j$ must be greater than $P_i$, and would have occurred earlier in the list. Since we assumed that $(t, P_i)$ is the first entry in the list that $x$ satisfies, it follows that $x$ does *not* satisfy any $c_j$ for $j$ not in $S_i$. Hence $x$ is labeled 1 in the mixture if and only if it is classified by some $c_j$ for $j \in S_i$. So the probability that it is labeled 1 is exactly the sum of the probabilities of these functions $c_j$, which is $P_i$. ■

## 4.2 LEARNING DISJUNCTIONS FROM A MIXTURE

Consider the 2-mixture MIX$(c_1, c_2, \nu)$ of monotone disjunctions. First, note that information-theoretically, it is impossible to learn when $\nu = 1/2$. For instance, suppose the example distribution $\mathcal{D}$ is such that only the two examples $\langle x_1 \rangle$ and $\langle x_2 \rangle$ occur, each with probability 50%. In this case, the mixture of $c_1 = x_1$ and $c_2 = x_2$ produces the same distribution on labels as the mixture of $c_1 = (x_1 \vee x_2)$ and $c_2 \equiv 0$. So without loss of generality, we will assume $\nu < 1/2$.

Recall that if $\mathbf{h}$ is the 2-p-DL with decreasing probabilities corresponding to this mixture, then we can express $c_1$ and $c_2$ in terms of the real-valued function $\mathbf{h}$: $c_1(x) = 1$ iff $\mathbf{h}(x) > 1/2$, and $c_2(x) = 1$ iff $\mathbf{h}(x) = 1$ or $\mathbf{h}(x) = \nu$. Kearns and Schapire [KS90] present an algorithm (which we call Learn-p-DL) that learns a $p$-DL that is an $\epsilon$-good model of probability of a target $p$-DL with decreasing probabilities. We show below how we can use their algorithm to solve our problem of learning monotone disjunctions from a mixture.

We claim that the following algorithm will PAC-learn both monotone disjunctions $c_1$ and $c_2$ from a mixture. Since it is not realistic to require that $\nu$ be precisely known, we will only assume that $\nu$ is bounded away from both 1/2 and 0.

**Algorithm** Learn-2-Mixture:

*Given: Access to an oracle MIX$(c_1, c_2, \nu)$, a value $\nu_b = 1/2-\gamma$ for some $\gamma > 0$, such that $\nu \in [\gamma, \nu_b]$, and $\epsilon, \delta > 0$. Let $\epsilon' = \min\{\gamma/2, \epsilon\}$.*

1. *Invoke* Learn-p-DL *with access to MIX, and parameters $(\epsilon', \delta)$, and obtain a p-DL $\mathbf{h}$.*
2. *Output $h_1 = \mathbf{h}$, changing entries $(t, r)$ to $(t, 1)$ for $r > 1/2$, and to $(t, 0)$ for $r < 1/2$.*
3. *Output $h_2 = \mathbf{h}$, changing entries $(t, r)$ to $(t, 1)$ when $r \in [1-\epsilon', 1] \cup [\epsilon', 1/2-\epsilon']$, and to $(t, 0)$ otherwise.*

**Theorem 5** *Algorithm* Learn-2-Mixture *PAC-learns monotone disjunctions from a 2-mixture MIX$(c_1, c_2, \nu)$ given a bound $\nu_b = 1/2-\gamma$ such that $\nu \in [\gamma, \nu_b]$, in time polynomial in $(1/\epsilon, 1/\delta, 1/\gamma)$.*

**Proof:** It is clear that the algorithm runs in time polynomial in $1/\epsilon, 1/\delta, 1/\gamma$. Let $\mathbf{l}$ be the p-DL corresponding to MIX. Consider hypothesis $h_1$. We claim it is an $\epsilon$-approximation to $c_1$. Suppose for some $x$, $h_1(x) \neq c_1(x)$. Then it must be the case that either $c_1(x) = 1$ and $h_1(x) = 0$, or $c_1(x) = 0$ and $h_1(x) = 1$. In other words, either $\mathbf{l}(x) \geq 1/2 + \gamma$ and $\mathbf{h}(x) \leq 1/2$, or $\mathbf{l}(x) \leq 1/2 - \gamma$ and $\mathbf{h}(x) \geq 1/2$. In both cases, $|\mathbf{h}(x) - \mathbf{l}(x)|$ is at least $\gamma > \epsilon'$, but the algorithm Learn-p-DL guarantees that the probability of this is at most $\epsilon$.

Let $R$ be the region $\{1\} \cup [\gamma, 1/2-\gamma]$ and $R'$ be the region $[1-\epsilon', 1] \cup [\epsilon', 1/2-\epsilon']$. Then if for some $x$, $h_2(x) \neq c_2(x)$, it must be that either $\mathbf{l}(x) \in R$ and $\mathbf{h}(x) \notin R'$ or $\mathbf{l}(x) \notin R$ and $\mathbf{h}(x) \in R'$. In both cases, $|\mathbf{l}(x) - \mathbf{h}(x)| > \epsilon'$, which can only happen with probability $< \epsilon$. ■

More generally, we show that the "majority concept" (if there is one) is learnable from a $k$-mixture, for the class of monotone disjunctions.

**Theorem 6** *The majority concept of a $k$-mixture of monotone disjunctions can be learned given a bound $\gamma > 0$ such that the probability associated with the majority concept is at least $1/2 + \gamma$. The running time*

*being polynomial in the usual parameters* $(1/\epsilon, 1/\delta, n)$ *and also in* $1/\gamma$.

**Proof:** Suppose $\mathbf{l}$ is the $k$-p-DL with decreasing probabilities corresponding to the mixture. Then, from the construction of the $k$-p-DL (Theorem 4), it is clear that for every entry $(t, r)$ such that $t$ satisfies the majority concept $c_1$, the $r$-value must be at least $p_1 \geq 1/2 + \gamma$. Also, for any term $t$ that does not satisfy $c_1$, the corresponding entry $(t, r)$ must have $r < 1/2$. This implies that for any $x$, $c_1(x) = 1$ exactly when $\mathbf{l}(x) > 1/2 + \gamma$. Invoke Learn-p-DL on this mixture, with parameters $(\min\{\gamma/2, \epsilon\}, \delta)$, to obtain a $k$-p-DL $\mathbf{h}$. Then by exactly the same argument as in Theorem 5, the projection of $\mathbf{h}$ is an $\epsilon$-approximation to $c_1$. ■

### 4.3 LEARNING WITH A MINIMUM DISAGREEMENT ORACLE

Let $MD(\mathcal{C})$ be an oracle that given a set of labeled examples finds a concept in class $\mathcal{C}$ with fewest disagreements on that sample. Angluin and Laird [AL88] show how one can use such an oracle to learn a class $\mathcal{C}$ in their random misclassification model with polynomial sample size. Sloan [Slo88] extends their result to the malicious misclassification model. Thus, Sloan's result immediately implies that given $MD(\mathcal{C})$ we can learn the "majority concept" in any $\mathrm{MIX}(c_1, c_2, \nu)$ for $c_1, c_2 \in \mathcal{C}$ and $\nu < 1/2$. We show here that if $\nu$ is also bounded away from 0, then we can learn the minority concept as well.

**Theorem 7** *Given access to an oracle* $MD(\mathcal{C})$ *and* $\nu_b \in (0, 1/2)$, *we can learn* $\mathrm{MIX}(\mathcal{C}, \nu_b)$ *in time polynomial in* $\frac{1}{\epsilon}, \frac{1}{\delta}, \frac{1}{\nu_b}, \frac{1}{1/2-\nu_b}$, *and* $\log(|\mathcal{C}|)$.

The proof is given in the appendix.

## 5 FINDING A CONSISTENT PAIR OF MONOTONE DISJUNCTIONS IS NP-HARD

A related problem to the mixture model is the following. Given an *arbitrary* set of labeled examples, can one determine whether there exist two concepts in a class $\mathcal{C}$ such that each example is consistent with at least one of them? One immediately notices that this is not so interesting, however, since the trivial concept pair TRUE and FALSE always satisfies this requirement. But, what if both concepts are required to be nontrivial? In that case, even for the simple class of monotone disjunctions, this problem becomes NP-hard. Note that this problem is not the same as asking whether there exists a 2-clause CNF formula consistent with a set of examples, since in that case we would require that all positive examples be positive for both clauses. In our problem, by contrast, we only require that each positive (negative) example be positive (negative) for at least one of the two disjunctions.

**Theorem 8** *The following decision problem is NP-complete: Given a set* $S$ *of* $N$ *example-label pairs, is there a pair of non-empty (i.e., not identically FALSE) monotone disjunctions* $c_1$, $c_2$, *such that for each* $(x, l) \in S$, *either* $l = c_1(x)$ *or* $l = c_2(x)$?

The proof appears in the appendix.

## 6 LEARNING CONCEPTS SWITCHED BY AN OBLIVIOUS ADVERSARY

In section 4, we showed how to efficiently learn in a situation where for each example, one of a small number of target concepts is chosen to classify it according to fixed probabilities, so long as the probability associated with one of the concepts is greater than 1/2. We now turn to the oblivious adversary switching model, which is easier in that the learner has some time-dependence information, but harder in that there may no longer be any "majority" concept.

Recall that in this model, an adversary specifies in advance, *before examples are drawn* from the distribution $\mathcal{D}$, a sequence $\{c_i\}$ of target concepts. Then, a sequence $\{x^i\}$ of examples is drawn from a distribution and each example $x^i$ is classified by the corresponding target concept $c_i$.

Note that a special case of an oblivious adversary is an adversary that switches according to a Markov process: For instance, in the case of two concepts, there is a certain probability $\beta < 1/2$ that the target concept switches from one example to the next, and upper and lower bounds on $\beta$ are available. This is an oblivious adversary since the probability $\beta$ is independent of the selection of examples from the distribution $\mathcal{D}$.

We will use the fact that if a sufficiently large sequence of $m$ examples is selected from $\mathcal{D}$, and a predetermined subsequence (not necessarily consecutive) of size $pm$ $(p < 1)$ is classified by $c_1$, then the empirical probability estimates used by the mixture algorithms are with high probability very close to those that would be obtained if the $m$ examples were taken from a mixture where the probability of classification $c_1$ is $p$. This will allow us to apply all the previous algorithms and results for mixtures to samples where a predetermined fraction of examples is classified by a particular concept.

We will only consider PAC-learning from an oblivious adversary switching between two monotone disjunctions $c_1$ and $c_2$.

If we take a large sample from the oracle, and it so happens that the fraction of $c_1-$classified examples is significantly different from 1/2, we can simply use the mixture algorithms to learn both concepts. However, when the $c_1-$fraction is close to 1/2, we need a more sophisticated strategy.

Our strategy is to filter out examples from a large se-

quence $S$ to create a sufficiently large new sample $S'$ such that: (a) The $c_1$−fraction in $S$ is significantly different from 1/2, and (b) The distribution $\mathcal{D}$ is not disturbed. We can then treat this new sample $S'$ as if it came from a mixture, and since it is a "good" sample, we can learn at least one of the concepts. To create $S'$, the algorithm guesses that a variable $x_i$ is relevant to one concept, say $c_1$, and not to the other. It then creates a sample $N_i$ of all negative examples with $x_i = 1$. If $x_i$ is indeed in $R(c_1) - R(c_2)$, all examples in $N_i$ must be $c_2$−classified. Unfortunately, $N_i$ is not a representative sample from $\mathcal{D}$ (it is biased toward $x_i = 1$), so we cannot learn a good approximation to $c_2$ from this sample. So, the algorithm creates a new sample $S_i$ consisting of the immediate successor of each example in $N_i$ in the original sequence $S$. Since examples are chosen randomly from a distribution, this "successor" sample $S_i$ is representative of $\mathcal{D}$. Also, since the average run length of any one concept is guaranteed to be be at least $1/\beta_u > 2$, we can expect that a significantly larger than 1/2 fraction of the examples in the sample $S_i$ will be $c_2$−classified. Thus $S_i$ will be a good sample, and the mixture algorithms can then be used to learn at least one of the concepts. The difficulty, of course, is that a variable $x_i \in R(c_1) - R(c_2)$ is not known. So the algorithm performs this procedure with all $n$ variables, and uses a test to determine which of the different $S_i$ samples is good.

Once an approximation $h_1$ is found to one of the concepts, say $c_1$, the algorithm takes a new, sufficiently large sample and filters out all examples $x$ for which $h_1(x) = 1$, so that (almost ) any example that is classified positive is $c_2$−classified. This fact enables the algorithm to learn a good approximation $h'$ to $c_2$ under the condition $h_1(x) = 0$. It also learns $h_{12}$, a good approximation to $c_1 \wedge c_2$ under the original distribution. Finally, it returns $h_{12} \vee (\neg h_1 h')$ as a good approximation to $c_2$.

We give our algorithm below, followed by a proof sketch. We use $m(\epsilon, \delta, n)$ to denote the number of examples of a 2-pdl c needed by Kearns and Schapire's Learn-p-DL algorithm, in order to produce with probability $(1 - \delta)$, a 2-pdl which is an $\epsilon$-good model of probability of c. $m$ is polynomial in $1/\delta, 1/\epsilon$, and $n$. If $S$ is a sample of $m_1$ examples classified by $c_1$ and $m_2$ examples classified by $c_2$, each selected according to $\mathcal{D}$, we call the 2-pDL $\mathrm{MIX}(c_1, c_2, m_2/(m_1 + m_2))$ the 2-pDL *associated with* $S$.

**Algorithm** Switch-Learn: *Given:* $\epsilon > 0, \delta > 0, \beta_l > 0, \beta_u < 1/2$. *Let* $\epsilon_1 = \min\{(1/2 - \beta_u)/8, \beta_l/2, \epsilon^2/200\}$, *and let* $\delta_1 = \delta/(10n)$

1. *Collect a sequence* $S$ *of* $\frac{4n}{3\beta_l\epsilon_1} m(\epsilon_1, \delta_1, n)$ *examples.*
2. *Run* Learn-p-DL *on* $S$ *to obtain a 2-pDL,* $\mathbf{h}$ *that is a* $\epsilon_1$*-good model of probability of the pDL associated with* $S$.
3. *Let* $h_{12} = \mathbf{h}$, *changing entries* $(t, r)$ *to* $(t, 0)$ *for* $r < (1 - \epsilon_1)$, *and all others to* $(t, 1)$. $h_{12}$ *is our hypothesis for* $c_1 \wedge c_2$.
4. *Take a random sample* $T$ *of* $(1/\epsilon_1^2)\log(1/\delta)$ *examples. Let* $q$ *be the number of examples* $x \in T$ *such that* $\mathbf{h}(x) \in [\epsilon_1, 1 - \epsilon_1]$. *If* $q/|T| < 5\epsilon_1$, *then return* $h_1 = h_2 = h_{12}$.
5. *Otherwise, for each* $x_i$, *let* $N_i = \{x \in S | x_i = 1,$ *and* $x$ *was classified negative*$\}$. *If for all* $i$, $|N_i| < \frac{3\beta_l\epsilon_1}{4n}|S|$ *then return FAILURE.*
6. *For each* $x_i$ *such that* $|N_i| \geq \frac{3\beta_l\epsilon_1}{4n}|S|$, *do the following:*
   (a) *Create a "successor" sample* $S_i$ *by including in* $S_i$ *all examples in the sequence* $S$ *that immediately follow an example in* $N_i$.
   (b) *Run* Learn-p-DL *on* $S_i$, *to obtain a 2-p-DL* $\mathbf{h}_i$ *that is an* $\epsilon_1$*-good model of* $S_i$.

   *Take a random sample* $T$ *of* $O(1/\epsilon_1 \log(n/\delta))$ *examples. Let* $\Delta = \frac{1}{2}(1/2 - \beta_u)$.

   *If for some* $i$, $\mathbf{h}_i(x) \in [1/2 - \Delta/2, 1/2 + \Delta/2]$ *occurs for a smaller than* $3\epsilon_1/2$ *fraction of* $x \in T$, *then let* $h_1 =$ *projection of* $\mathbf{h}_i$. *Otherwise return FAILURE.*
7. *Take a sample* $S_1$ *of* $(10/\epsilon)m(\epsilon_1, \delta_1, n)$ *examples. Let* $S'$ *be the set of* $x \in S_1$ *such that* $h_1(x) = 0$. *If* $|S'| < (\epsilon/10)|S_1|$, *then return* $h_1$, *and* $h_2 = h_{12}$.
8. *Otherwise, run* Learn-p-DL *with error parameter* $\epsilon_1$ *on* $S'$, *to obtain a 2-pdl* $\mathbf{h}'$. *Let* $h' = \mathbf{h}'$, *changing entries* $(t, r)$ *to* $(t, 1)$ *if* $r > \epsilon_1$, *and to* $(t, 0)$ *otherwise. Return* $h_1$, *and* $h_2 = h_{12} \vee (\neg h_1 h')$.

**Theorem 9** *For any oblivious adversary with parameters* $\beta_l$ *and* $\beta_u$ *switching between two monotone disjunctions, for any* $\epsilon, \delta > 0$, *algorithm* Switch-Learn, *PAC-learns both disjunctions, in time polynomial in* $1/\epsilon, 1/\delta, 1/\beta_l$, *and* $1/(1/2 - \beta_u)$

**Proof Sketch:** We make use of the fact that all samples which are given to the Learn-p-DL algorithm are sufficiently large, so that they can be treated as if they came from some mixture of $c_1$ and $c_2$. Let c be the 2-pDL associated with the sample $S$ in step 1. We are guaranteed by the definition of an oblivious adversary that the fractions of examples classified by $c_1$ and $c_2$ are at least $\beta_l$, so that $(c_1 \wedge c_2)(x) = 1$ iff $\mathbf{c}(x) = 1$, and $(c_1 \wedge c_2)(x) = 0$ iff $\mathbf{c}(x) < 1 - \beta_l$. Because $\epsilon_1 \leq \beta_l/2$, this implies that in step 3, $h_{12}$ is an $\epsilon_1$-approximation of $c_1 \wedge c_2$ since $h_{12}(x) \neq (c_1 \wedge c_2)(x)$ would imply that $|\mathbf{h}(x) - \mathbf{c}(x)| > \epsilon_1$, which occurs in at most an $\epsilon_1$ portion of the distribution.

In step 4, we test whether there is a significant portion of the distribution for which $c_1(x) \neq c_2(x)$. Note that $c_1(x) \neq c_2(x)$ exactly when $\mathbf{c}(x) \in [\beta_l, 1 - \beta_l]$. By Chernoff bounds, if $\mathbf{Pr}_x[\mathbf{h}(x) \in [\beta_l/2, 1 - \beta_l/2]] \geq 6\epsilon_1$, then with high probability $\mathbf{h}(x) \in [\epsilon_1, 1 - \epsilon_1]$ occurs for a smaller than $5\epsilon_1$-fraction of the examples in $T$.

So, if the observed fraction is less than $5\epsilon$, we can assume the above probability is at most $6\epsilon_1$. Since $\mathbf{h}$ is an $\epsilon_1(\leq \beta_l/2)$-good model of probability of $\mathbf{c}$, this implies that $\mathbf{c}(x) \in [\beta_l, 1-\beta_l]$ occurs with probability at most $7\epsilon_1$. Thus $c_1(x)$ and $c_2(x)$ differ on at most $7\epsilon_1$ portion of the distribution $\mathcal{D}$. So by our choice of $\epsilon_1$, $h_{12}$ is an $\epsilon$-approximation to both $h_1$ and $h_2$.

If the fraction $q/|T|$ is at least $5\epsilon_1$ then by a similar argument we know with high probability that there is at least a probability $3\epsilon_1$ that $c_1(x) \neq c_2(x)$.

If the algorithm reaches step 5, therefore, we may assume without loss of generality, $\mathbf{Pr}_x[c_1(x) = 1 \text{ and } c_2(x) = 0] \geq 3\epsilon_1/2$. So for some variable $x_i \in R(c_1) - R(c_2)$, $x_i = 1$ and $c_2(x) = 0$ on at least $3\epsilon_1/(2n)$ portion of the distribution $\mathcal{D}$. Since the definition of an oblivious adversary guarantees that each concept occurs least $\beta_l$ fraction of the time, it follows that over the sample $S$, with high probability, we will see at least an $3\beta_l\epsilon_1/(4n)$ fraction of negative examples that have $x_i = 1$. Thus in step 5, at least one of the sets $N_i$ will be large enough. (The chance of returning FAILURE in that step will be very small) Also, this means that in step 6, for at least one of the $x_i$, all the negative examples in $N_i$ are $c_2$-classified (since $x_i \in R(c_1) - R(c_2)$). Each time we select an example to be classified by $c_2$, there is a fixed probability that it will belong to $N_i$. Since we are guaranteed that at most a $\beta_u$ fraction of $c_2$-classified examples have their successors classified by $c_1$, with high probability at most a $(\frac{1}{2}+\beta_u)/2$ fraction of the examples in $S_i$ are $c_1$-classified. So, the sample $S_i$ will be a "good" sample.

The algorithm now tests to see which $S_i$ is a "good" sample in that the fraction of examples classified by each $c_j$ is sufficiently far from $1/2$. It does this by first learning a 2-p-DL $\mathbf{h}_i$ from the sample $S_i$. To see that the test works, suppose the $c_1$-fraction in $S_i$ is between $1/2 - \Delta/4$ and $1/2 + \Delta/4$, where $\Delta = \frac{1}{2}(1/2 - \beta_u)$. Then the pDL $\mathbf{c}$ associated with the sample will have the property that $\mathbf{c}(x) \in (1/2-\Delta/4, 1/2+\Delta/4)$ iff $c_1(x) \neq c_2(x)$. At this point in the algorithm, we know that $\mathbf{Pr}_x[c_1(x) \neq c_2(x)] \geq 3\epsilon_1$, so $\mathbf{c}(x) \in (1/2 - \Delta/4, 1/2 + \Delta/4)$ on at least $3\epsilon_1$ portion of the distribution. Now, the 2-pdl $\mathbf{h}$ is an $\epsilon_1$-good model of $\mathbf{c}$, so for at least $2\epsilon_1$ portion of the distribution $\mathcal{D}$, $\mathbf{h}(x)$ must be in the interval $(1/2-\Delta/2, 1/2+\Delta/2)$, since otherwise $|\mathbf{c}(x) - \mathbf{h}(x)| > \Delta/4 > \epsilon_1$ occurs with probability (over $\mathcal{D}$) more than $\epsilon_1$. Thus if in step 6, out of $O(\frac{1}{\epsilon_1}\log(1/\delta))$ random examples we see less than a $3\epsilon_1/2$ fraction of examples $x$ such that $\mathbf{h}(x) \in [1/2 - \Delta/2, 1/2 + \Delta/2]$, with high probability $S$ does not contain greater than $(1/2-\Delta/4)$ fraction of the minority concept. The hypothesis $h_1$ will be then be an $\epsilon_1$-approximations to, say, $c_1$ (as argued for mixtures).

Conversely, if a sample $S_i$ is a good sample, i.e., contains at most a $1/2-\Delta = (\frac{1}{2}+\beta_u)/2$ fraction classified by the minority concept, then the corresponding 2-pDL $\mathbf{c}$ has no entries at probabilities strictly between $1/2-\Delta$ and $1/2+\Delta$. Since $\mathbf{h}_i$ is an $\epsilon_1$-good approximation of $\mathbf{c}$, the chance that $\mathbf{h}_i(x)$ is in the interval $[1/2 - \Delta/2, 1/2 + \Delta/2]$ is at most $\epsilon_1$, so the test will with high probability produce less than a $3\epsilon_1/2$ fraction of examples $x$ for which $\mathbf{h}(x) \in [1/2 - \Delta/2, 1/2 + \Delta/2]$. Thus the chance that the algorithm returns FAILURE in step 6 is very small.

We now have found the first concept. Steps 7 and 8 deal with finding the second.

In step 7, the algorithm takes a new large sample $S_1$ and removes all examples $x$ for which $h_1(x) = 1$. If $\mathbf{Pr}_x[h_1(x) = 0] \geq \epsilon/5$, then with high probability at least an $\epsilon/10$ fraction of examples will remain. So, if fewer than that many are seen, we may assume $h_1(x) = 0$ on at most an $\epsilon/5$ portion of the distribution. Since $h_1$ is an $\epsilon_1 \leq \epsilon/5$-good approximation of $c_1$, this also implies that the probability that $c_1(x) = 0$ is at most $2\epsilon/5$. Thus $h_{12}$ is an $\epsilon$-approximation to $c_2$.

If the algorithm does not stop in Step 7, then similarly we may assume that the probability of $h_1(x) = 0$ is at least $\epsilon/20$. Since $h_1$ is an $\epsilon^2/200$-approximation to $c_1$, this implies that the conditional probability $\mathbf{Pr}_x[c_1(x) = 1 | h_1(x) = 0] \leq \epsilon/10$. Since $\mathbf{h}'$ is an $\epsilon_1$-good model of the underlying p-DL, the probability that $h'(x) \neq c_2(x)$ (under $h_1(x) = 0$) is at most

$$\begin{aligned}
&\mathbf{Pr}[c_1(x) = 1 | h_1(x) = 0] + \\
&\mathbf{Pr}[h'(x) = 1, c_1(x) = 0, c_2(x) = 0 \mid h_1(x) = 0] + \\
&\mathbf{Pr}[h'(x) = 0, c_2(x) = 1 \mid h_1(x) = 0] \\
&\leq \epsilon/10 + \epsilon_1 + \epsilon_1.
\end{aligned}$$

So, $h'$ is a $3\epsilon/10$-close approximation to $c_2$ under $h_1(x) = 0$. It is then easy to show that this implies that $h_{12} \vee (\neg h_1 \wedge h')$ is an $\epsilon$-approximation to $c_2$ under the original distribution $\mathcal{D}$. ■

# 7 WHEN IS LEARNING A MODEL OF PROBABILITY AS EASY AS LEARNING A DECISION RULE?

In addition to the notion of learning with a model of probability (see Section 2), Kearns and Schapire [KS90] define a weaker notion of learning a $p$-concept class *with a decision rule*. A "decision rule" is a standard $\{0,1\}$-valued concept. Let $R_{\mathcal{D}}(\mathbf{c}, h)$ be the probability that decision rule $h$ misclassifies an example $x$ chosen from $\mathcal{D}$ and labeled according to $\mathbf{c}$. An algorithm learns a $p$-concept class $\mathcal{C}$ with a decision rule if for all $\mathbf{c} \in \mathcal{C}$, distributions $\mathcal{D}$, and $\epsilon, \delta > 0$, with probability $1-\delta$ the algorithm outputs a decision rule $h$ such that $R_{\mathcal{D}}(\mathbf{c}, h) \leq R_{\mathcal{D}}(\mathbf{c}, \pi_{\mathbf{c}}) + \epsilon$. (Recall, $\pi_{\mathbf{c}}$ is the projection of $\mathbf{c}$.) We call such an $h$ an *$\epsilon$-good* decision rule. Kearns and Schapire note that if one can (polynomially) learn $\mathcal{C}$ with a model of probability, then one can (polynomially) learn $\mathcal{C}$ with a decision rule by producing the projection of the learned model of probability (with some appropriate changes to $\epsilon$). We examine here the question: under what conditions does the converse

hold?

We use the language of mixtures to provide sufficient conditions. Say that $\mathcal{C}$ is *closed under mixture* with $\mathcal{C}'$ if for any $\mathbf{c} \in \mathcal{C}, \mathbf{c} \in \mathcal{C}'$, and $\nu \geq 0$, the $p$-concept $\mathrm{MIX}(\mathbf{c}, \mathbf{c}', \nu)$ is in $\mathcal{C}$.[4] In particular, we will be concerned with classes closed under mixture with $\{T, F\}$.

**Theorem 10** *Let $\mathcal{C}$ be closed under mixture with $\{T, F\}$. If $\mathcal{C}$ is (polynomially) learnable with a decision rule, then $\mathcal{C}$ is (polynomially) learnable with a model of probability.*

The condition of being closed under mixture with $\{T, F\}$ is satisfied by many natural $p$-concept classes such as $p$-DL's (with decreasing probabilities) and the "nondecreasing functions" mentioned in [KS90]. Before proving Theorem 10, let us first state a simple lemma.

**Lemma 11** *If $h$ is an $\epsilon$-good decision rule and $\alpha > 0$, then:*

$$\mathbf{Pr}_{x \in \mathcal{D}}\left[\mathbf{c}(x) \notin [\tfrac{1}{2} - \tfrac{\alpha}{2}, \tfrac{1}{2} + \tfrac{\alpha}{2}] \text{ and } h(x) \neq \pi_{\mathbf{c}}(x)\right] \leq \epsilon/\alpha.$$

**Proof:** Suppose $x$ is such that $\mathbf{c}(x) \notin [\frac{1-\alpha}{2}, \frac{1+\alpha}{2}]$ and $h(x) \neq \pi_{\mathbf{c}}(x)$. Then the probability the label assigned by $\mathbf{c}$ is $h(x)$ is at most $\frac{1}{2} - \frac{\alpha}{2}$, whereas the probability the label equals $\pi_{\mathbf{c}}(x)$ is at least $\frac{1}{2} + \frac{\alpha}{2}$. So, the difference is at least $\alpha$. Thus, if the probability of this event exceeds $\epsilon/\alpha$, then $h$ is not an $\epsilon$-good decision rule. ■

For $p$-concept $\mathbf{c}$ and probability $p$, let us define $\pi_{\mathbf{c},p}(x)$ to equal 1 if $\mathbf{c}(x) \geq p$ and 0 otherwise.

**Proof of Theorem 10:** We are given $\epsilon$ and $\delta$ and for convenience reduce $\epsilon$ if necessary so that $2/\epsilon$ is integral. Let $A$ be the algorithm to learn $\mathcal{C}$ with a decision rule. The idea of the conversion is to use $A$ to create $2/\epsilon$ decision rules $h_0, h_1, \ldots, h_{(2/\epsilon)-1}$, where $h_i$ is an approximation to $\pi_{\mathbf{c},\epsilon i/2}$. The model of probability $\mathbf{h}$ will be the function $\mathbf{h}(x) = \epsilon i/2$ where $i$ is the greatest index such that $h_i(x) = 1$. Note that if $\mathbf{c}(x) = p$ then $\pi_{\mathbf{c},q}(x) = 1$ for all $q \leq p$. So, if each $h_i$ actually were equal to $\pi_{\mathbf{c},\epsilon i/2}$, then for all $x$, $|\mathbf{h}(x) - \mathbf{c}(x)|$ would be at most $\epsilon/2$.

Concept $h_0$ is simple: $h_0 = T$. For $i > 1$, define $p_i = \epsilon i/2$ and $q_i = \frac{1}{2} - |p_i - \frac{1}{2}|$. Define $p$-concept $\mathbf{c}_i$ to be $\mathrm{MIX}(\mathbf{c}, T, 1 - \frac{1}{2q_i})$ if $p_i \leq 1/2$, and $\mathrm{MIX}(\mathbf{c}, F, 1 - \frac{1}{2q_i})$ if $p_i > 1/2$. These quantities are chosen so that if $\mathbf{c}(x) = p_i + \alpha$ ($\alpha$ need not be positive), then $\mathbf{c}_i(x) = 1/2 + \alpha/2q_i$, as can be seen by the following calculation. If $p_i \leq 1/2$, so $p_i = 1 - q_i$, then for $\mathbf{c}(x) = p_i + \alpha$ we have $\mathbf{c}_i(x) = (1 - q_i + \alpha)\frac{1}{2q_i} + 1(1 - \frac{1}{2q_i}) = \frac{1}{2} + \frac{\alpha}{2q_i}$. If $p_i > 1/2$, then $p_i = q_i$, so $\mathbf{c}_i(x) = (q_i + \alpha)\frac{1}{2q_i} = \frac{1}{2} + \frac{\alpha}{2q_i}$. Note that we can run algorithm $A$ on target concept $\mathbf{c}_i$ by feeding to $A$ examples $x$ chosen from $\mathcal{D}$, with the classification assigned by $\mathbf{c}$ replaced with probability $1 - \frac{1}{2q_i}$ by 1 or 0 respectively depending on whether $p_i \leq 1/2$ or $p_i > 1/2$.

For each $i > 0$, let $h_i$ be the outcome of running $A$ on $\mathbf{c}_i$ as described above, with confidence parameter $\delta\epsilon/2$ and error parameter $\epsilon^3/4$. The confidence parameter implies that with probability at least $1 - \delta$ all the $h_i$ are $\epsilon^3/4$-good rules, and let us assume that this is indeed the case. Define $E_i(x)$ to be the event that $\mathbf{c}_i(x) \notin [1/2 - \epsilon/4, 1/2 + \epsilon/4]$ and $h_i(x) \neq \pi_{\mathbf{c}_i}(x)$. So, by Lemma 11, for each $i$ we have $\mathbf{Pr}_{x \in \mathcal{D}}[E_i(x)] \leq \epsilon^2/2$, which implies that with probability at least $1 - \epsilon$, none of the events $E_i(x)$ occur. By our previous calculation, if $\mathbf{c}(x) \notin [p_i - \epsilon/2, p_i + \epsilon/2]$ then $\mathbf{c}_i(x) \notin [1/2 - \frac{\epsilon}{4q_i}, 1/2 + \frac{\epsilon}{4q_i}]$, which implies $\mathbf{c}_i(x) \notin [1/2 - \epsilon/4, 1/2 + \epsilon/4]$. This means we must have $h_i(x) = \pi_{\mathbf{c}_i}(x)$ for $E_i(x)$ not to occur. So, with probability at least $1 - \epsilon$, for each $i$ such that $p_i < \mathbf{c}(x) - \epsilon/2$ we have $h_i(x) = 1$ and for each $i$ such that $p_i > \mathbf{c}(x) + \epsilon/2$ we have $h_i(x) = 0$. Since there is some $p_i$ between $\mathbf{c}(x) - \epsilon$ and $\mathbf{c}(x) - \epsilon/2$, with probability at least $1 - \epsilon$ the greatest index $i$ such that $h_i = 1$ is between $\mathbf{c}(x) - \epsilon$ and $\mathbf{c}(x) + \epsilon/2$. Thus, $\mathbf{h}$ is an $\epsilon$-good model of probability. ■

[4] A mixture of two $p$-concepts is the obvious extension of a mixture of two "standard" concepts. With probability $1 - \nu$, example $x$ is labeled 1 with probability $\mathbf{c}(x)$, and with probability $\nu$, it is labeled 1 with probability $\mathbf{c}'(x)$.

## 8 REMARKS AND OPEN PROBLEMS

In the strong-adversary switching model, we believe we also have a "2-competitive" algorithm for learning with queries when the adversary may switch between 3 disjunctions. That is, the number of mistakes plus queries made is at most a fixed polynomial plus 2 times the number of switches. We conjecture that a "$(k-1)$-competitive" strategy is possible when there are $k$ disjunctions.

The problem of learning disjunctions (in polynomial time) in Sloan's malicious misclassification model remains open. One intermediate goal might be to learn the majority concept in a mixture of $n$ disjunctions, where the majority concept has probability $p > 1/2 + \alpha$ for some known $\alpha > 0$.


### Acknowledgments

We would like to thank Michael Kearns, Rob Schapire, Sridhar Tayur, and Manfred Warmuth for helpful discussions. Avrim would like to acknowledge support from an NSF Postdoctoral Fellowship.


## References


[AL88] Dana Angluin and Philip Laird. Learning from noisy examples. *Machine Learning*, 2(4):343–370, 1988.

[ALRS] S. Ar, R.J. Lipton, R. Rubinfeld, and M. Sudan. Reconstructing algebraic func-

tions from mixed data. Unpublished Manuscript.

[BDBK+90] S. Ben-David, A. Borodin, R.M. Karp, G. Tardos, and A. Wigderson. On the power of randomization in on-line algorithms. In *Proc. 22nd Annual ACM Symposium on Theory of Computing*, pages 379–388, 1990.

[BLS87] A. Borodin, N. Linial, and M. Saks. An optimal online algorithm for metrical task systems. In *Nineteenth Annual ACM Symposium on Theory of Computing*, pages 373–382, 1987.

[HL91] D. Helmbold and P. Long. Tracking drifting concepts using random examples. In *Proceedings of the Fourth Annual Workshop on Computational Learning Theory*, pages 13–23, Santa Cruz, California, August 1991. Morgan Kaufmann.

[KL88] Michael Kearns and Ming Li. Learning in the presence of malicious errors. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, pages 267–280, Chicago, Illinois, May 1988.

[KS90] Michael J. Kearns and Robert E. Schapire. Efficient distribution-free learning of probabilistic concepts. In *Proceedings of the Thirty-First Annual Symposium on Foundations of Computer Science*, pages 382–391. IEEE, 1990.

[KSS92] Michael J. Kearns, Robert E. Schapire, and Linda M. Sellie. Toward efficient agnostic learning. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, 1992.

[Lev91] E. Levin. Modeling time variant systems using hidden control neural architecture. In *Neural Inf. and Proc. Systems 3 (NIPS)*, 1991.

[MMS90] M.S. Manasse, L.A. McGeoch, and D.D. Sleator. Competitive algorithms for on-line problems. *Journal of Algorithms*, 11:208–230, 1990.

[Riv87] Ronald L. Rivest. Learning decision lists. *Machine Learning*, 2(3):229–246, 1987.

[Slo] R.H. Sloan. personal communication.

[Slo88] Robert H. Sloan. Types of noise in data for concept learning. In David Haussler and Leonard Pitt, editors, *First Workshop on Computational Learning Theory*, pages 91–96. Morgan Kaufmann, August 1988.

# APPENDIX

**Proof of Theorem 7:** Let $c_1, c_2$ be the two target concepts where $\nu = \mathbf{Pr}(c_2) \in [\frac{1}{2} - \nu_b, \nu_b]$. Sloan [Slo88] shows we can $\epsilon$-approximate $c_1$, and we can make the error rate of the hypothesis so small that with high probability no difference between it and $c_1$ is seen in the following discussion. Therefore, we may assume for convenience that we have $c_1$ exactly (this makes the algorithm only polynomial in $1/\delta$ and not $\log(1/\delta)$). For any two concepts $c$ and $c'$, let $SD(c, c')$ be the symmetric difference of $c$ and $c'$, and let $d(c, c')$ be the probability measure of $SD(c, c')$. Let $d = d(c_1, c_2)$ and let $m$ be some sufficiently large quantity polynomial in the parameters given in the statement of the theorem.

Sample from $\mathcal{D}$ until a set $S$ of $2m$ examples $(x, l)$ have been found such that $l \neq c_1(x)$. If $d > \epsilon$ then with high probability we will have found $2m$ such examples after $4m/(\epsilon(\frac{1}{2} - \nu_b))$ selections from $\mathcal{D}$, so if we do not find $2m$ we can output $c_1$ as a hypothesis for $c_2$. Now, select a set $T$ of $m$ new examples from $\mathcal{D}$. Finally, let $h_2$ be the output of $MD(\mathcal{C})$ on sample set $S \cup T$. We claim that with high probability, $h_2$ is $\epsilon$-close to $c_2$.

Concept $c_2$ is correct on all examples from $S$, and with high probability is incorrect on at most $[d(1-\nu)+\gamma]m$ of the examples from $T$ for small constant $\gamma$ (Hoeffding bounds). Now, let $h$ be a hypothesis with error greater than $\epsilon$ with respect to $c_2$, and let $\epsilon_1$ be the measure of that error in $SD(c_1, c_2)$ and let $\epsilon_2$ be the measure in $X - SD(c_1, c_2)$; so $\epsilon_1 + \epsilon_2 = \epsilon$. So, we expect $h$ to make $2m(\epsilon_1/d) \geq 2m\epsilon_1$ errors on set $S$ and $m[\nu(\epsilon_1 + \epsilon_2) + (1-\nu)(d - \epsilon_1 + \epsilon_2)]$ errors on set $T$. Rearranging terms, we have:

$$\begin{aligned} &\mathbf{E}(\text{number of errors of } h) \\ &\geq m[d(1-\nu) + (\epsilon_2 - \epsilon_1) + 2\nu\epsilon_1 + 2\epsilon_1] \\ &\geq m[d(1-\nu) + \epsilon_1 + \epsilon_2] \\ &= m[d(1-\nu) + \epsilon]. \end{aligned}$$

With high probability, for $\gamma < \epsilon/2$ and $m$ sufficiently large, the number of errors is greater than the error rate of $c_2$.

So, with high probability any given hypothesis $h$ with $d(h, c_2) > \epsilon$ will have more disagreements on $S \cup T$ than $c_2$. For sufficiently large $m$, the probability is so close to 1 that with high probability all such hypotheses in $\mathcal{C}$ will have more disagreements than $c_2$ (i.e., the standard Occam argument). Thus, with high probability we learn $c_2$. ■

**Proof of Theorem 8:** For convenience, we identify an example with the set of variables that are assigned 1 in the example. For instance, $\langle x_i, x_j \rangle$ denotes an example $x$ that sets the variables $x_i$ and $x_j$ to 1, and sets all others to 0. Also, $(\langle x_1, x_j \rangle, l)$ denotes an example-label pair, where $l \in \{-, +\}$.

First, we note that two monotone disjunctions $c_1$, $c_2$ are consistent (in that at least one classifies each example

correctly) with a set $S$ of example-label pairs, if and only if the following two conditions hold. First, for each positive example $(x, +)$, at least one variable set to 1 in $x$ is relevant to one of $c_1$ or $c_2$. Second, for each negative example $(x, -)$ either all variables set to 1 in $x$ are irrelevant to $c_1$, or all variables set to 1 in $x$ are irrelevant to $c_2$.

We exploit these constraints in reducing 3-SAT to this problem. Consider any 3-CNF formula $F$ over the variables $v_1, v_2 ... v_m$. With each variable $v_i$ we associate two variables $x_i, y_i$, and with each negated variable $\neg v_i$, we associate the two variables $x_{m+i}, y_{m+i}$.

For each variable $v_i$, we create the following four labeled examples:

(A) $( \langle x_i, x_{m+i} \rangle, + )$

(B) $( \langle y_i, y_{m+i} \rangle, + )$

(C) $( \langle x_i, y_i \rangle, - )$

(D) $( \langle x_{m+i}, y_{m+i} \rangle, - )$

In addition, for $i \neq j (i, j \in \{1, 2, ..., m\})$, we create the following negative examples:

(E) $(\langle x_i, x_j \rangle, -), \quad (\langle x_i, x_{m+j} \rangle, -), \quad (\langle x_{m+i}, x_j \rangle, -), \quad (\langle x_{m+i}, x_{m+j} \rangle, -)$

(F) $(\langle y_i, y_j \rangle, -), \quad (\langle y_i, y_{m+j} \rangle, -), \quad (\langle y_{m+i}, y_j \rangle, -), \quad (\langle y_{m+i}, y_{m+j} \rangle, -)$

Finally, for each clause in the formula $F$, we create a positive example containing $x_i$ if $v_i$ is in the clause, and $x_{m+i}$ if $\neg v_i$ is in the clause. Thus for a clause $(v_1 \vee (\neg v_3) \vee v_4)$, we create the positive example $(\langle x_1, x_{m+3}, x_4 \rangle, +)$.

It is easy to see that if there is a satisfying assignment $a$ for $F$, then the following two disjunctions will be consistent with the examples constructed:

$$c_1 = ( \bigvee_{a(v_i)=1} x_i ) \vee ( \bigvee_{a(v_i)=0} x_{m+i} ),$$

$$c_2 = ( \bigvee_{a(v_i)=1} y_{m+i} ) \vee ( \bigvee_{a(v_i)=0} y_i ).$$

For the other direction, assume that the set of examples is consistent with two non-empty monotone disjunctions $c_1$ and $c_2$. Let $R(c_1)$ and $R(c_2)$ denote the corresponding sets of relevant variables.

We claim that only one of the sets $R(c_1)$ or $R(c_2)$ contains $x$-variables. Consider a particular $x_i$, say $x_1$. Since $\langle x_1, x_{m+1} \rangle$ is a positive example (A above), at least one of $\{x_1, x_{m+1}\}$ must belong to at least one of $R(c_1)$ or $R(c_2)$. Without loss of generality, say $x_1 \in R(c_1)$. Then $R(c_2)$ cannot contain any $x_j$ or $x_{m+j}$ for $j \neq 1$, since the examples $\langle x_1, x_j \rangle$ and $\langle x_1, x_{m+j} \rangle$ are negative examples (E). Next, suppose $R(c_2)$ contains one of $x_1$ or $x_{m+1}$. Then because of the negative examples (E), none of the variables $x_j$ or $x_{m+j}$ for $j \neq 1$ can belong to $R(c_1)$. Thus the variables $x_j, x_{m+j}$ do not belong to either $R(c_1)$ or $R(c_2)$. However, the positive examples (A) make this impossible, so it cannot happen that $R(c_2)$ contains $x_1$ or $x_{m+1}$. Thus only $R(c_1)$ contains $x$-variables, without loss of generality.

Consistency with the positive examples (A) requires that for each $i$, at least one of $\{x_i, x_{m+i}\}$ belong to $R(c_1)$. We claim that $R(c_1)$ cannot contain both $x_i$ and $x_{m+i}$, for any $i$. Suppose the contrary, i.e., that $R(c_1)$ contains both $x_i$ and $x_{m+i}$, for some $i$. Then the negative examples (C) and (D) enforce that neither $y_i$ nor $y_{m+i}$ belong to $R(c_2)$. However, since $\langle y_i, y_{m+i} \rangle$ is a positive example (B), at least one of $\{y_i, y_{m+i}\}$ must belong to $R(c_1)$. Then because of the negative examples (E) and (F), none of the variables $\{x_j, x_{m+j}, y_j, y_{m+j}\}$, for $j \neq i$ can belong to $R(c_2)$. Also, as argued above, $R(c_2)$ cannot contain any $x$-variables. Thus $R(c_2)$ is empty, which we assumed is not the case. So our assumption that $R(c_1)$ contains both $x_i$ and $x_{m+i}$ must be wrong.

Thus without loss of generality, $R(c_1)$ contains exactly one of $\{x_i, x_{m+i}\}$ for each $i$, and $R(c_2)$ does not contain any $x$-variables. Since the positive examples corresponding to the clauses contain only $x$-variables, it must be the case that for each such positive example, at least one variable assigned to 1 in that example must belong to $R(c_1)$.

Then a satisfying assignment for $F$ is defined by $a(v_i) = 1$ if $x_i \in R(c_1)$, and $a(v_i) = 0$ if $x_{m+i} \in R(c_1)$. ■