When to say "Don't Know": Confidence in Automatically Generated Hypotheses without the Assumption of an Underlying Distribution

Iain Morrow
Department of Mathematics, London School of Economics
Houghton Street, London WC2A 2AE, United Kingdom
email: iain.morrow@gmail.com

05-December 2005
CDAM Research Report LSE-CDAM-2005-17

Abstract

We have a set $S \subset \{0,1\}^n$, together with, for each $x \in S$, the result of some unknown function $F : \{0,1\}^n \to \{0,1\}$ applied to x, and a method for generating a hypothesis $h \in H$ about F given S. We present theoretical and experimental results on four possible methods (similarity, convexification, prevalence and Hamming distance) for determing, given two elements $y,z \in \{0,1\}^n \backslash S$, whether we should be more confident that $h(y) = F(y)$, or that $h(z) = F(z)$, or indeed that we should attach the same degree of confidence to both statements. We consider whether it is possible to have an absolute measure of confidence in the statement that $h(a) = F(a)$ for any given $a \in \{0,1\}^n \backslash S$. We introduce a modification of a standard learning algorithm for Boolean functions, which naturally partitions new examples into three categories: 1,0 and don't know.

1 Introduction

The theory of learning Boolean functions has been significantly developed over the last few years, particularly within the framework of the *probably approximately correct* (PAC) model proposed by Valiant (1984). In this framework, the learner is given a set $S \subset \{0,1\}^n$ (the *training sample*), together with, for each $x \in S$, the result of some unknown function $F : \{0,1\}^n \to \{0,1\}$ applied to x. It processes this information to generate a hypothesis h about F. Crucially, the PAC model assumes that the elements of S are drawn at random from the set of all possible *examples* (i.e. all Boolean strings of length n) based on some fixed but unknown probability distribution (say μ), and that any *new* examples that are shown to the learner for it to classify using h are drawn from all possible examples based on the same distribution μ. Under these assumptions, Valiant showed that for a broad range of function classes, a learner can, given a large enough training sample S, efficiently generate a hypothesis h about F that, with any chosen probability less than 1, will agree with F on $(100-\varepsilon)\%$ of new randomly drawn examples, for any desired $\varepsilon > 0$.

Investigating the question of what the learner should do if the assumption of a fixed distribution μ is *not* valid is the main aim of this paper. The fact that a new example is drawn based on the same distribution as the original examples that the learner was trained on means that in some sense the new examples are quite like the training sample, and what Valiant showed was that this relationship allows us to make precise statements about how likely our hypothesis is to be correct on a new example. This paper considers some other ways of determining how like the original examples a new example is, in the absence of a common distribution, and asks whether any of them provide us with a way to give a degree of confidence to our classification. We look at both the theoretical basis of these methods (Section 4), and at how they perform in practice (Section 7), using standard datasets from the Machine Learning Database held at the University of California at Irvine (Hettich et al. 1998). We also present a number of algorithms that use the new measures (in particular, similarity as proposed by Anthony & Hammer (2004)) to estimate confidence (Section 6), to generate hypotheses (Section 5.1) and to identify likely outliers in datasets (Section 5.2).

Another reason for considering alternative ways of determining the relationship between a new example and the examples used for training is that even in existing frameworks like the PAC framework, we may need a significant number of examples (possibly more than are available) for the confidence bounds provided by that framework to be useful. We explore this in more detail later, but we first briefly cover the terminology and notation we use in this paper when discussing Boolean functions. Readers familiar with Boolean functions can skip the next section.

2 Learning Boolean Functions

In this paper, we are interested in producing a hypothesis h about an unknown Boolean function $F : \{0,1\}^n \to \{0,1\}$; we assume that h is also a Boolean function. Elements of $\{0,1\}^n$ are referred to as *examples*, and there are two types of example, as defined below:

Definition 2.1 *Positive/ Negative Example*
$x \in \{0,1\}^n$ *is a* positive example *of the function* F *if* $F(x) = 1$.
$x \in \{0,1\}^n$ *is a* negative example *of the function* F *if* $F(x) = 0$.

Each $x \in \{0,1\}^n$ can be thought of as an n-tuple $(x_1, x_2, \ldots, x_n)$. We refer to the element x_i as *bit* i *of* x. As well as the whole space $\{0,1\}^n$, we are sometimes interested in subsets T of the space, where each member of T has certain bits fixed, and the other bits are allowed to vary; these are called *subcubes*. We first define the concept of a *positional substring*, and then the definition for subcube follows naturally. A positional substring of x is a series of bits from x, together with the positions of those bits. For example, suppose $x = 11010$. Then $*1**0$ is a positional substring of length 2 (where $*$ is a placeholder that can represent either 0 or 1). Notice that the position of the specified bits is crucial; $*1**0 \neq *10**$, for example. Formally:

Definition 2.2 *Positional Substring*
Let $I \subseteq [n] = \{1, 2, \ldots, n\}$, *with* $|I| = k$. *Let* $y = x|_I$ *be the bits of* x *in the positions corresponding to the elements of* I. *Then* (y, I) *is a positional substring of* x.

Definition 2.3 *Subcube*
Let (y, I) *be any positional substring. Then* $T = \{x \in \{0,1\}^n : x|_I = y\}$ *is a* subcube *of* $\{0,1\}^n$. *We say that* T *has co-dimension* $k = |I|$ *and dimension* $n - k$.

2.1 Disjunctive Normal Form

Consider the simple Boolean function which gives 1 on $x \in \{0,1\}^n$ if and only if a given bit of x is also 1. Call this function u_i, where i is the position of the bit in x. Define $\overline{u}_i$ to be the *complement* of this function, i.e. $\overline{u}_i(x) = 1 \Leftrightarrow u_i(x) = 0$. These functions are known as *literals*, and functions of the form $\overline{u}_i$ are known as *negated* literals.

Now consider the Boolean functions formed by the *conjunction* of one or more literals. These functions are known as *monomials*. For example, the function $u_1 \wedge \overline{u}_2 \wedge u_3$ is a Boolean function which returns 1 on x if and only if the first bit of x is 1 AND the second bit is 0 AND the third bit is 1. It is usual, when writing monomials, to leave out the conjunction signs (so the example given earlier would be written as $u_1\overline{u}_2u_3$). We also note that a monomial with no negated literals (i.e. which does not contain any $\overline{u}_i$) is called a *monotone* monomial.

There is a simple one-to-one relationship between monomials and positional substrings. Given a positional substring (y,I), with $I = \{i_1, i_2, \ldots, i_k\}$ and $y = \{y_1, y_2, \ldots, y_k\}$, consider the monomial m formed by the conjunction of the literals u_{i_j} (where $y_j = 1$) and negated literals $\overline{u}_{i_l}$ (where $y_l = 0$). Then for $x \in \{0,1\}^n$, $m(x) = 1$ iff x contains the positional substring (y,I). For example, the monomial corresponding to the positional substring $(1011, \{3,5,6,7\})$ is $u_3\overline{u}_5u_6u_7$. We will frequently use this relationship, speaking of the monomial "corresponding to" a positional substring, and vice versa.

We now go one stage further and consider Boolean functions formed by the *disjunction* of one or more monomials:

Definition 2.4 *Disjunctive Normal Form*
Suppose we have a function $F : \{0,1\}^n \to \{0,1\}$. Then F can be written as $m_1 \vee m_2 \vee \ldots \vee m_l$ where m_i is a monomial and such a representation is called a Disjunctive Normal Form.

It is well known that any Boolean function F has a (not necessarily unique) representation in Disjunctive Normal Form. We say that $F \in D_{n,k}$ if we can write F as a disjunction of monomials m_i such that the maximum number of literals in any m_i is $\leq k$. Clearly $\forall k$, $D_{n,k} \subseteq D_{n,k+1}$ and $D_{n,n}$ is the set of all functions $F : \{0,1\}^n \to \{0,1\}$.

We also consider later the *number* of terms in a DNF of F.

Definition 2.5 *l-term k-DNF*
F is an l-term k-DNF if F has a DNF with at most l terms, where each term has at most k literals. We write $F \in D^l_{n,k}$

We are interested here in learning (i.e. producing a hypothesis) from *data*, in particular from an initial set of examples called a training sample:

Definition 2.6 *Training Sample*
A training sample *S for a function t is a set of ordered pairs $(x_i, t(x_i))$ where $x_i \in \{0,1\}^n$.*

2.2 Existing Frameworks for Learning Boolean Functions

There are several existing frameworks for learning Boolean functions. In the discussion that follows, we will refer to *learning algorithms*, which are simply functions that take as input a training sample for a given unknown function $t : X \to \{0,1\}$ and output a hypothesis $h : X \to \{0,1\}$ where X is the example space ($\{0,1\}^n$ for our purposes).

2.2.1 Probably Approximately Correct Learning

Valiant (1984) gave the first definition of probably approximately correct, or *PAC*, learning. A learning algorithm is said to be PAC if:

Definition 2.7 *PAC learning algorithm*
Let L be a learning algorithm and let $h = L(S)$ be the hypothesis that it produces given the training sample S. Let $er_\mu(h)$ be the probability that the hypothesis h disagrees with the target concept t on an example drawn at random according to μ and $\mathbb{S}(m,t)$ be the set of training samples of size m for the target concept t. Let $\mu^m(\Theta)$ be the probability that a sample of size m drawn from $\mathbb{S}$ according to μ is in Θ. Then L is PAC *if:*

$$\forall \varepsilon > 0, \forall \delta > 0, \exists m_0 = m_0(\delta, \varepsilon) \ \ s.t. \ \ \forall t, \forall \mu, \forall m > m_0, \ \ \mu^m\{S \in \mathbb{S}(m,t) : er_\mu(L(S)) < \varepsilon\} > 1 - \delta$$

Informally, "the probability that a randomly chosen training sample will lead to a hypothesis which has error $< \varepsilon$ is $> 1 - \delta$".There are many known PAC learning algorithms, and in particular any learning algorithm for functions on finite Boolean spaces that is *consistent* is PAC (for a proof of this fact see for example Anthony & Biggs (1992)).

Definition 2.8 *Consistency*
A training sample S is consistent *for a given hypothesis space H if there is some element $h_0 \in H$ such that $\forall s \in S$, $h_0(s) = t(s)$, i.e. one of the possible hypotheses agrees with the target concept on all the labelled examples in the training sample.*

A learning algorithm L that produces hypotheses in H is consistent if given a training sample S that is consistent with H, $h = L(S) \in H$ satisfies $h(s) = t(s)$, $\forall s \in S$, where t is the unknown target function. In other words, h classifies each element of the training sample in the same way as the target function.

Valiant showed that we can efficiently learn certain types of target function, including DNFs with most k literals in each term, for some fixed $k < n$. Valiant's algorithm is consistent (for a proof of this see Anthony & Biggs), and therefore PAC. We present a modified version in Section 6.

Consideration of PAC learning immediately raises the question of how big m_0 has to be to guarantee PAC learning. Bounds on m_0 are known[1]:

$$m_0 \leq \frac{1}{\varepsilon} \log_2 \left(\frac{|\, H \,|}{\delta} \right)$$

where H is the (finite) hypothesis space. In this paper it is generally assumed to be $D_{n,k}$ for some fixed $k < n$, and it can easily be shown that $|\, D_{n,k} \,| \leq 2^{(2n)^k}$. Hence by the bound stated above we can, with any consistent DNF learning algorithm, generate a PAC hypothesis with a sample of size no more than:

$$\frac{1}{\varepsilon} \log_2 \left(\frac{|\, D_{n,k} \,|}{\delta} \right) \leq \frac{(2n)^k}{\varepsilon} \log_2 \left(\frac{1}{\delta} \right)$$

[1] for a longer explanation including proofs see Anthony & Biggs (1992)

We can also give a lower bound on the size of the sample we need, using the *Vapnik-Chervonenkis dimension* [2] of $D_{n,k}$. It can be shown that the Vapnik-Chervonenkis dimension of $D_{n,k}$, $VCdim(D_{n,k}) \geq \binom{n}{k}$ (see Anthony & Biggs page 114). It can also be shown that any PAC learning algorithm needs a sample of size at least:

$$\frac{VCdim(H) - 1}{32\varepsilon}$$

Consideration of these bounds shows that even for relatively simple hypothesis classes we may not be able to guarantee PAC learning with a sample of any reasonable size. We might therefore want to be cautious and investigate other confidence measures before making predictions, even in the PAC framework.

2.2.2 Reliable and Useful Learning

The *reliable and useful learning* approach introduced by Rivest & Sloan (1988) shows that it is possible to learn Boolean functions in such a way that the output hypothesis either gives the correct answer or outputs "don't know" (in other words it never wrongly classifies an example). As with PAC learning, examples are assumed to be drawn according to a fixed probability distribution.

Rivest & Sloan also made the assumption that rather than having the algorithm try to produce a hypothesis for the target concept "all at once", it would be taught sub-concepts first and these would be used to build up the hypothesis for the concept as a whole. More precisely, the teacher splits the target concept into a number of sub-concepts (and possibly sub-sub-concepts and so on), each of which is a simple disjunction or conjunction of two of the inputs or of two lower level concepts. The learner starts by requesting a labelled example, and the teacher provides this example, labelled according to the first low level concept. The learner asks for more examples until it has learnt that concept to its satisfaction. When it next requests an example, the teacher classifies the example according to the next low level concept. This process continues until all the sub-concepts (and the concept of interest) have been covered. For example, to learn the function $u_1u_2 \vee u_3u_4$ the learner would first be taught u_1u_2 (call this s_1) then u_3u_4 (call this s_2) and then $s_1 \vee s_2$. This is illustrated by Figure 1 on page 7.

However, Kivinen (1995) showed that without the teacher breaking up the target concept as assumed by Rivest & Sloan, it is not possible to reliably and usefully learn even monotone monomials over $\{0,1\}^n$ in time polynomial in n. This suggests that finding an absolute measure of confidence that can be computed efficiently is likely to be difficult at best.

[2] For a treatment directly relevant to computational learning theory, see again Anthony & Biggs

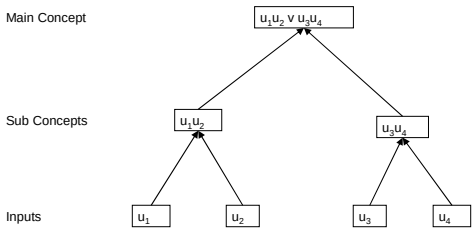


Figure 1: Example of a simple disjunction of two short monomials, broken down into sub-concepts as required for reliable and useful learning

2.2.3 Transductive Confidence Machines

We also briefly note that Vovk (2002) has proposed a learning method which he calls a *Transductive Confidence Machine*. We do not go into details of the method here, except to note that it is designed to produce a hypothesis which both classifies new examples and assigns a measure of confidence to that classification. While this is clearly similar to the subject of this paper, Vovk's work again assumes that examples are drawn at random according to some fixed but unknown probability distribution μ. In this paper we are looking at what can be said without the assumption of a fixed underlying distribution, and we will not discuss Vovk's work further.

3 Dealing with Numeric Data

Our focus in this paper is learning Boolean functions, but of course much real world data does not come in this form. We can use the results presented here for data in numeric form, through a process called *binarization*; see Boros et al. (1996) for a more complete explanation.

Suppose that we have a set $A \subseteq \mathbb{R}^n$ of numeric examples, and suppose that for each example $a \in A$ we know whether a is a positive example or a negative example. This gives us n numeric inputs, which we can convert to a set of corresponding Boolean inputs as follows. Consider a single numeric input $i = 1, \ldots, n$. Take the value of this input in each example $a \in A$ and call the set of all these values $\Gamma_i = \{\gamma_{i1}, \gamma_{i2}, \ldots\}$. We call the γ_{ij} the *cut points*. We then introduce a set of Boolean binarization functions b_{ij}, defined as:

$$b_{ij}(x) = \begin{cases} 1 & if\ x \geq \gamma_{ij} \\ 0 & otherwise \end{cases}$$

We repeat this process for each numeric input i, and then compute, for each resulting b_{ij} function, the value of $b_{ij}(a_i)$ for each $a \in A$, where a_i is the i^{th} input for example a. These values are the initial set of binary inputs (and there are ij of them for each example).

Clearly there can be a very large number of these functions b_{ij}, but it turns out that we can discard

most of them as unnecessary. The first observation is that only cut-points that separate positive and negative examples are required. We call these *essential* cut-points.

The second observation is that what we need is a set of inputs that is small yet still separates the positive and negative examples in our training set; Boros et al. call this a *support set*. We find a support set as follows.

Suppose we have q essential cut-points. For each cut-point, we have a corresponding binary input whose value is given by the appropriate b_{ij} function from earlier, acting on the appropriate numeric input. Suppose we are dealing with medical data, and one of the numeric inputs is the patient's temperature. We have found that 38°C is an essential cut-point. Then the corresponding binary input would be "temperature above 38°C", and if the value of this input was 1 for a given patient, it would indicate that the patient was running a fever.

Let $C = \{c_p\}$ be the set of binary inputs corresponding to the essential cut-points. Each example $a \in A$ now corresponds to an element of $\{0,1\}^q$. Let B be the set of binarized versions of all examples in A. Partition B into two samples, one containing all the positive examples B^+ and one containing all the negative examples B^-. Then, for each $y \in X = \{(x^+, x^-) : x^+ \in B^+, x^- \in B^-\}$, determine the binary inputs on which the two elements of y differ. Suppose there are $l = |B^+||B^-|$ such pairs. Define indicator functions I_{rp}, with $I_{rp} = 1$ if pair r of examples differs in bit p ($r = 1,\ldots,l$ and $p = 1,\ldots,q$) and $= 0$ otherwise. We then need to find a subset Z of the binary inputs C such that, for each pair X, at least one of the binary inputs on which they differ is included in Z, i.e. $\forall r = 1,\ldots,l$, $\exists c_p \in Z : I_{rp} = 1$. This means that each positive example will differ from any negative example in at least one bit, using the reduced set of binary inputs Z. We find Z by solving the following integer programme:

$$minimise \sum_{k=1,\ldots,q} y_k \; subject\ to \sum_{j=1,\ldots,q} y_j I_{ij} \geq \lambda, \forall i = 1,\ldots,l$$

where $y_k = 1$ if c_k is included in Z, and 0 otherwise. Note that $\sum_k y_k = |Z|$. Minimising this sum keeps the support set as small as possible, which speeds up later analysis. Note also that the constraints are defined in terms of a parameter λ. This represents the desired minimum separation between any positive and any negative example. It is normally set to 1, but can be increased if we want to achieve a more robust separation. Of course, increasing λ will increase the eventual size of Z, which will increase the time taken by the learning algorithm, and the time taken for the binarization. For the experimental results presented in this paper we have always taken $\lambda = 1$.

We now turn to solving the integer programme. Boros et al. note that this problem is equivalent to set-covering, which is known to be NP-hard, but that a simple greedy algorithm (details can be found in their paper) will produce an approximate solution in polynomial time. Broadly, their algorithm adds at each stage the binary input that increases the total separation of the positive and negative examples (summed over all pairs not yet distance λ apart) by the largest amount. Using the approx-

imate solution from the greedy algorithm means that Z will be larger than absolutely necessary, but experimental results with standard datasets suggest that the support sets generated are of a reasonable size (and there may be less danger of over-fitting the binarization to the training sample). For the datasets used in this paper, which each contain about 10 numeric inputs, we find that a support set contains between 12-16 inputs, which agrees well with the binarization of some of the same datasets by Hammer & Bonates (2005).

4 Confidence Measures - Definitions and Theoretical Relationships

We now turn to the main focus of this paper, namely possible confidence measures usable without requiring the assumption of a fixed distribution for the examples. The intuitive notion underlying our discussion of confidence measures is that the more a new example y has in common with members of the training sample S, the more likely we are to be able to classify it correctly. We give four possible ways to measure how much y has in common with the members of S: Hamming distance, prevalence, convexity and similarity. We look here at how the measures are related in theory, and then in Section 7 use them with standard data sets to see how they perform in practice.

4.1 Hamming Distance

One obvious way to measure how closely y is related to the members of S is to ask how far y is from S using the well-known *Hamming distance* measure. Let $d(x,y)$ be the Hamming distance between $x,y \in \{0,1\}^n$. We can then define $\mathbb{D}_{k,A} = \{w \in \{0,1\}^n : d(w,A) \leq k\}$, the set of points Hamming distance $\leq k$ from A. Clearly $\mathbb{D}_{0,A} = A, \mathbb{D}_{j,A} \subseteq \mathbb{D}_{j+1,A}, \mathbb{D}_{n,A} = \{0,1\}^n$. We consider minimum Hamming distance from the sample as a measure of confidence, and show in Section 7 that our hypotheses do seem to be more accurate on examples which are a low Hamming distance from our training sample.

4.2 Prevalence of Monomials

Suppose we have a hypothesis $h \in D_{n,k}$ for some $k < n$, and a training sample S split into a set of positive examples S^+ and a set of negative examples S^-, i.e. $S = S^+ \cup S^-$. We are presented with two new examples $x_1, x_2 \in \{0,1\}^n$. If $h(x_i) = 1$, there must be at least one monomial in h which gives 1 on x_i. Call the set of these monomials M_i. Now if h is consistent, each $m_i \in M_i$ must be satisfied by $N \geq 0$ positive examples in the training sample, and no negative examples. Using a medical analogy for a moment, suppose h is our hypothesis about the symptoms that indicate a particular disease. Then for a new patient z, $h(z) = 1$ means that h classifies z as having the disease in question because z exhibits a set of symptoms that have not been seen in any patient who doesn't have the disease, and have been seen in N patients known to have the disease (or if $N = 0$, have not been seen in any patients at all).

We would expect to be increasingly confident in our diagnosis as N increases, and conversely if N is 'small', we would be cautious; in the extreme case $N = 0$, it would be difficult to justify diagnosing the disease. We would also expect to be increasingly confident in our diagnosis as the *proportion* of positive examples that satisfy elements of M_i increases. Suppose for example we can find a monomial m in h which is satisfied by both x_1 and 50 positive examples in the training sample, but that there is no monomial m' in h which satisfies both x_2 and any set of 10 or more positive examples in the training sample. We might intuitively expect that the classification of x_1 as a positive example is more likely to be correct, as there is more "supporting evidence" from S^+.

Based on this intuition, and following Hammer & Bonates (2005), we say that the *prevalence* of a monomial m is the proportion of positive examples s in the training set for which $m(s) = 1$. We introduce the closely related concept of a *prevalence count* ρ for each monomial m, where ρ is defined as the number of positive examples in our training sample (i.e. elements of S^+) on which m gives 1. More formally:

Definition 4.1 *Prevalence*
The prevalence count of m is $\rho(m,S) = |\{s \in S^+ : m(s) = 1\}|$
The prevalence of m is: $\frac{\rho(m,S)}{|S^+|}$
m is k-prevalent on A if $\rho(m,S) \geq k$.

Since h is a disjunction of a finite number of monomials, we can define for each example classified as positive by our hypothesis (i.e. $\forall x \in \{0,1\}^n$ with $h(x) = 1$) the maximum prevalence count over all the monomials in the DNF of h that x satisfies (i.e. $m_i : m_i(x) = 1$). Call this function $p(x,h,S)$.

Definition 4.2 *Maximum Positive Prevalence of x*
Let $h = \vee_i m_i$. *The Maximum Positive Prevalence of x is* $p(x,h,S) = max_{i:m_i(x)=1}\rho(m_i,S)$

We can then construct the sets $\mathbb{P}_k(h,S) = \{x \in \{0,1\}^n, h(x) = 1 : p(x,h,S) \geq k\}$. Clearly $\forall i$, $\mathbb{P}_{i+1}(h,S) \subseteq \mathbb{P}_i(h,S)$. We propose that as i increases, we should be more confident in the classification $h(x) = 1$. Experimental results in Section 7 suggest that this is correct up to a point. More precisely, they suggest that there is a threshold prevalence count, such that monomials with a prevalence count below this are not accurate in classifying new examples. Monomials with a prevalence count above this threshold are substantially more accurate but the accuracy does not increase with increasing prevalence count once we have passed the threshold.

4.3 Convexification

It is well known that Hamming distance is a metric, and so $\forall x,y,z \in \{0,1\}^n$, $d(x,z) \leq d(x,y) + d(y,z)$. Suppose we say that y is *between* x and z in the case where this is actually an equality. Equivalently,

consider the subcube $T \subseteq \{0,1\}^n$ which has x and z as its opposite corners. Then the points between x and z are precisely the elements of that subcube.

Having defined what *between* means in $\{0,1\}^n$, we can, given a set A, define $\mathbb{C}_i(A)$, the *i-convexification* of A, to be the set of all elements of $\{0,1\}^n$ that are between two elements x,y of A, where the Hamming distance between x and y is no more than i. We can think of i-convexification as "filling in" all subcubes of co-dimension at most i which have two elements of A at opposite corners.

Definition 4.3 *i-convexification*
$\mathbb{C}_i(A) = \{w \in \{0,1\}^n : \exists x,y \in A \text{ with } d(x,w) + d(w,y) = d(x,y) \leq i\}$.

Clearly $A \subseteq \mathbb{C}_i(A)\ \forall i$, and $A \subseteq \mathbb{C}_2(A) \subseteq \mathbb{C}_3(A) \ldots \subseteq \mathbb{C}_n(A) \subseteq \{0,1\}^n$. Note that the last of these, $\mathbb{C}_n(A)$ is not necessarily equal to the whole of $\{0,1\}^n$; there may be points in $\{0,1\}^n$ that are not between any points in A. Intuitively, we might expect that elements of $\{0,1\}^n$ that are in $\mathbb{C}_i(A)$ for i "small" are more like points in A than those that are not.

Rather than the slightly clumsy statement "w is in some convexification of A", we propose the term *flanked* and define the *flanking distance* $\mathbb{F}(w,A)$. Formally:

Definition 4.4 *Flanked*
We say that w is flanked by A if $\exists i$ such that $w \in \mathbb{C}_i(A)$, and we write $\mathbb{F}(w,A) = i$ if $w \in \mathbb{C}_i(A)$ and $w \notin \mathbb{C}_j(A), \forall j < i$. Further we can say that w is closely flanked *by A if i is "small".*

While in this paper we will focus on the convexification hierarchy defined above (which we will call the *one application convexification hierarchy*), Anthony & Hammer (2005) have proposed two other convexification hierarchies. First, consider what happens if we apply i-convexification to A to produce $\mathbb{C}_i(A)$, and then apply i-convexification to $\mathbb{C}_i(A)$ and so on, repeating this process until the resulting set does not change under i-convexification. Call this final set $\mathbb{C}_i^*(A)$, and the process *iterated i-convexification*. Then we have the hierarchy:

Definition 4.5 *Iterated i-convexification hierarchy*
$A \subseteq C_2^*(A) \subseteq C_3^*(A) \ldots \subseteq C_n^*(A) \subseteq \{0,1\}^n$

The set $\mathbb{C}_i^*(A)$ is also known as the *i-convex hull of A*. Note that $C_n^*(A)$ may not be the whole of $\{0,1\}^n$.

Secondly, start with A and apply 2-convexification to get $\mathbb{C}_2(A)$, which we will call $\mathbb{I}_2(A)$ then apply 3-convexification to $\mathbb{C}_2(A)$ to get $\mathbb{I}_3(A)$ and so on. This gives the hierarchy:

Definition 4.6 *Increasing convexification hierarchy*
$A \subseteq \mathbb{I}_2(A) \subseteq \mathbb{I}_3(A) \ldots \mathbb{I}_n(A) \subseteq \{0,1\}^n$

Again, $\mathbb{I}_n(A)$ may not be the whole of $\{0,1\}^n$.

Experimental results on the single application convexification hierarchy, using binarizations of standard datasets, are given in Section 7. We note that when we experimented with iterated convexification we found that, for the datasets used, $\mathbb{C}_2^*(A)$ is either the whole of $\{0,1\}^n$ or very close to it, which means that the hierarchy is trivial. This may well be at least partly an artifact of the binarization process. Valiant's $D_{n,k}$ learning algorithm takes $O(mn^k)$ time, where m is the size of the training sample, n is the number of binary inputs and k is the maximum length of any monomial in the hypothesis. Since this time increases like the k^{th} power of n, the binarization process is designed to keep n small, which means that the Hamming distance between examples in the training sample will be small.

4.3.1 Computing Convexification Hierarchies

Algorithms for computing convex hulls in $\mathbb{R}^n$ are known e.g. QuickHull (Barber, Dobkin & Huhdanpaa 1996), but these are not applicable to Boolean spaces. Ekin et al. (1998) provide a polynomial time algorithm for finding the *k-convex hull* of $A \subseteq \{0,1\}^n$, which in our notation is $\mathbb{C}_k^*(A)$. Ekin's algorithm works by considering the DNF of the Boolean function F which takes the value 1 on all points $a \in A$ and the value 0 elsewhere, then repeatedly searching for monomials m_1, m_2 in the DNF which are distance $\leq k$ apart. If such monomials are found, the algorithm removes them and replaces them with a new monomial which includes precisely those literals found in both m_1 and m_2. The process is repeated until there are no such pairs of monomials, at which point each monomial is distance at least $k+1$ from any other. The first stage of this algorithm takes $O(|A|^2)$ time, since all possible pairs of monomials have to be checked. Subsequent stages will take less time, as we remove monomials from the DNF at each stage.

However, we are more often interested in the result of applying a single *k-convexification* operation to a set $A \subseteq \{0,1\}^n$ rather than finding the k-convex hull. We present Algorithm 4.7 on page 13, which computes the k-convexification hierarchy of a set $A \subseteq \{0,1\}^n$ with each set in the hierarchy taking $O(|A|\binom{n}{k})$ time, which is faster than Ekin et al. for $|A| > \binom{n}{k}$. Further, experimental evidence suggests that $\mathbb{C}_k^*(A)$ is rather uninteresting in practice (in that $\mathbb{C}_k^*(A) = \{0,1\}^n$ for all or almost all possible k), whereas the convexification hierarchy generated by the single application of k-convexification does seem to be useful in estimating confidence.

To understand why the iterated convexification hierarchy might be uninteresting in practice, we first note that for the special case $k = n$, we can give a simple and rather weak necessary and sufficient condition for the convex hull of a set of points A to be the whole of $\{0,1\}^n$, which is that $\forall i = 1,\ldots,n$ we can find $x^+, x^- \in A$ such that $x_i^+ = 1$ and $x_i^- = 0$. To show this, suppose we compute the set $\mathbb{C}_n^*(A)$. Let Γ be the DNF of the function that gives 1 on x if $x \in \mathbb{C}_n^*(A)$ and 0 otherwise. Now Γ is an $n-$*convex* function, and Ekin et al. showed that it must have a DNF consisting of at most 1 monomial (to see this, consider that if the DNF contained 2 monomials, they must be distance $\leq n$ apart and

so we could further convexify the function using the procedure suggested by Ekin et al.). Suppose $\Gamma = m$, say. Now m must contain at least one literal, which without loss of generality we suppose to be u_1. Now there is by assumption $y \in A$ such that $y_1 = 0$, and $d(y,m) \leq n$ since we are in $\{0,1\}^n$. So following Ekin et al. we can replace m with the convex hull of y and m, which certainly has fewer literals in it than m, since we have removed u_1. So we have expanded the set of points covered by Γ by convexification. But by definition $\mathbb{C}_n^*(A)$ cannot be expanded by convexification. Hence our assumption that there were any monomials in the DNF of Γ must be false, and it must be the whole of $\{0,1\}^n$. This shows that the condition is sufficient. To see that it is necessary, simply note that any point in the convex hull of a set of points all of which have the same value for a particular bit must itself have the same value for that bit.

Suppose for $k < n$ we strengthen the condition in the obvious way to be: $\forall i = 1,\ldots,n$ we can find $x^+, x^- \in A$ such that $x_i^+ = 1$ and $x_i^- = 0$, with $d(x^+,x^-) \leq k$. It turns out that this condition is not always sufficient for large n (Appendix A gives a counter-example in $\{0,1\}^{20}$, with $k = 2$). However, we conjecture that there is some function β such that for $n \leq \beta(k)$, this condition is sufficient to imply that $\mathbb{C}_k^*(A) = \{0,1\}^n$.

Algorithm 4.7 *k-Convexification Algorithm*

In this algorithm $\mathbb{SP}_j(a)$ is the set of all points distance exactly j from the point $a \in \{0,1\}^n$, the j-sphere *around a.*

Set $C = A$ initially .
for j=2 to k
    Compute $\mathbb{SP}_j(0)$
    for each $x \in A$
        Exclusive-OR each element of $\mathbb{SP}_j(0)$ with x to produce the set $\mathbb{SP}_j(x)$.
        for each $y \in \mathbb{S}_j(x)$
            If $y \in A$, add all points between x,y to C
        end for
    end for
    Output C as $\mathbb{C}_j(A)$.
end for

Each iteration of the *for* loop compares each element $a \in A$ with every point in $\mathbb{SP}_j(a)$ which clearly has size $\binom{n}{j}$ and so generating each set in the convexification hierarchy takes $O(|A|\binom{n}{j})$ time. Clearly for the whole hierarchy, the running time will be $O(|A| \sum_{j=2}^{k} \binom{n}{j})$, and for small A, a naive algorithm that compared every element of A with every other would be faster [3].

[3] We can determine whether $y \in A$ in constant time, provided that A is constructed as a *hashtable* - see for example Cormen, Leiserson & Rivest (1990)

4.4 Similarity

Similarity was proposed by Anthony & Hammer (2004). The similarity of a binary string $x \in \{0,1\}^n$ to a set A is defined as the maximum k such that all *positional substrings* (recall Definition 2.2 on page 3) of length $\leq k$ in x are found in one or more members of A. This allows us to define similarity as follows:

Definition 4.8 *For $A \subseteq \{0,1\}^n$ and $x \in \{0,1\}^n$, the* similarity $s(x,A)$ *of x to A is the largest integer $k \geq 0$ such that for every positional substring (v,I) with $\mid I \mid = k$, $\exists y \in A$ such that (v,I) is also a positional substring of y, i.e:*

$s(x,A) = max\{i : \forall I \subseteq [n], |I| \leq i, \exists y \in A, y \mid_I = x \mid_I\}$

To illustrate the definition, suppose the set A consists of the following members of $\{0,1\}^6$:

$$\begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

and let $x = (011100)$. Then $s(x,A) \geq 2$, since all positional substrings of x of length ≤ 2 are contained in some element of A (not necessarily the *same* element). For example the positional substring $*1***0$ is contained in 110000 which is in A. But the positional substring $011**$ is not contained in any element of A, which implies that $s(x,A) < 3$, and so $s(w,A) = 2$.

Similarity can also be thought of as related to the presence or absence of symptoms in a group of patients. Consider a doctor who has seen many patients, some of whom have disease X and some who do not. The doctor bases his or her diagnosis on the presence of syndromes (combinations of symptoms) in the patient. If we suppose there are n possible symptoms, and identify each patient with $z \in \{0,1\}^n$, where bit i of z is set to 1 if the symptom is present and 0 otherwise, then a syndrome corresponds to a positional substring of z. Suppose a new patient arrives, and call the binary string corresponding to their symptoms z_α. If there is a positional substring β of length l contained in z_α such that the doctor has not previously seen any patient with that grouping of symptoms, the doctor might be reluctant to make a diagnosis if l was "small". In the extreme case where $l = 1$, the patient has a symptom that the doctor has never seen before, and it would be strange to make a diagnosis in this case. This is analogous to saying that if a new example has low similarity to the set of previously seen examples, we should be reluctant to classify that example. Clearly as l increases, we might become more confident. We note that we will always (unless the new string is *identical* to one seen

previously) have $s(z_\alpha, A) < n$, since any substring of length n is equal to the whole string, z_α, i.e. $s(x,A) < n \Leftrightarrow x \notin A$.

Based on this similarity measure, we define, for any $A \in \{0,1\}^n, k \in \mathbb{Z}^+$ the set of points with similarity at least k to A, $A_k^S = \{w \in \{0,1\}^n : s(w,A) \geq k\}$. Clearly $A = A_n^S \subseteq A_{n-1}^S \ldots \subseteq A_0^S = \{0,1\}^n$.

4.4.1 Computing Similarity

Anthony & Hammer note that for arbitrary w and A the problem is equivalent to set-covering, which is known to be NP-hard. However, they also showed that if we only want to know whether similarity is above a certain fixed $k \in \mathbb{Z}^+$, we can do this in time $O(|A|n^k)$, i.e. in time polynomial in n. The algorithm, which is essentially Valiant's DNF learning algorithm, is as follows:

Algorithm 4.9 *Compute Similarity*

Gener ate all possible monomials of length k.
for each $a \in A$
    remove any monomials that give 1 on a.
end for
Call the disjunction of any remaining monomials g_k

Then $s(w,A) \geq k \Leftrightarrow g_k(w) = 0$, since the monomials in g_k correspond precisely to those positional substrings of length k that do not appear in any element of A.

4.4.2 Implications of Similarity for Confidence

What are the implications of the similarity measure for confidence in our hypothesis h about a target concept t? Recall the notion of a l-term-k-DNF function, written $D_{n,k}^l$ (Definition 2.5 on page 4). Suppose we know that $h \in D_{n,a}^b, t \in D_{n,c}^d$ for some known $a,b,c,d \in \mathbb{Z}$. Veal (2005a) has shown that, if h is consistent with t and $j = max(a+d, b+c)$, $\exists \phi_m \in D_{n,j}$ such that $\phi_m(x) = 1 \Leftrightarrow h(x) \neq t(x)$. In other words, there is a function whose terms are of bounded size which will give 1 whenever the hypothesis and the target concept disagree (and only when they disagree). We might call $\phi_m(x)$ the misclassification function. The knowledge that this function exists allows Veal to show that $\forall x \in A$, whenever $s(x,A) \geq max(a+d, b+c)$, $h(x) = t(x)$, i.e. that the hypothesis will be correct on x.

Veal also showed that we cannot simply assume that a hypothesis is more likely to be correct on examples with high similarity to the training sample. He gives a set $A \subseteq \{0,1\}^4$, together with a target concept c and a hypothesis h, both in $D_{4,3}$, where h is consistent on A, such that all the elements of $\{0,1\}^4$ that are misclassified by h have *higher* similarity to A than those examples that are classified correctly. However, the set A is "unusual" in the sense that the *pervasiveness* of A is 0. Anthony & Hammer (2004) define pervasiveness as:

Definition 4.10 *The* pervasiveness *of A, written* $P(A) = min_{x \in \{0,1\}^n} s(x,A)$

The pervasiveness of A is linked to the size of the largest subcube of $\{0,1\}^n$ that does not have a member of A in it; we can find a subcube of $\{0,1\}^n$ of dimension $n-p-1$ that does not intersect A, but there is no subcube of dimension $n-p$ that does not intersect A.

We can show with a simple combinatorial argument that sets with pervasiveness 0 are rare. To say that $P(A) = 0$ implies that there is a dimension $n-1$ subcube $T \subset \{0,1\}^n$ which does not intersect A, i.e. that we can find $b \in \{0,1\}$ and $i \in \{1,\ldots,n\}$ such that $T = \{x \in \{0,1\}^n : x_i = b\}$. But this is the same as saying that there is some $j \in 1,\ldots,n$ such that $\forall a \in A$, $a_j = (1-b)$. Assume that each possible element of A is chosen independently from $\{0,1\}^n$ with probability p. Then $E(|A|) = p2^n$. For any given index j, the probability that all elements of A will agree on it is $\leq 2(\frac{1}{2}^{|A|}) = 2^{-p2^n+1}$. So the probability that the elements of A will agree on *any* index is $\gamma \leq 2n(\frac{1}{2^{|A|}}) = n2^{-p2^n+1}$, which is very small even for small n and A. For example, suppose $|A| = 10$ and $n = 5$. Then $\gamma \leq 2 * 5 * \frac{1}{2}^{10} < 0.01$. Further, Veal showed that as $n \to \infty$, almost surely we have that the pervasiveness of a randomly chosen set A tends to $n - \lfloor log_2(nlog_{\frac{1}{1-p}}(2)) \rfloor - 1$. It is unclear whether higher similarity implies higher accuracy for more "typical" values of pervasiveness.

4.5 Relationship between Similarity and Hamming Distance

Anthony & Hammer (2004) showed that making a statement about similarity is in a sense stronger than a statement about Hamming distance. They showed that if we can put a lower bound on the similarity of a point x to a set A we can put an upper bound on the Hamming distance between them. We present their result with a new proof:

Theorem 4.11 *Suppose for* $x \in \{0,1\}^n, A \subseteq \{0,1\}^n$ *we have* $s(x,A) \geq k$. *Then* $d(x,A) \leq n-k$. *Hence* $A_k^s \subseteq \mathbb{D}_{n-k,A}$.

Proof. Suppose $s(x,A) \geq k$. Let (z,I) be a positional substring of x with $\mid I \mid = k$. Then $\exists y \in A$ such that (z,I) is also a positional substring of y. But this means that x and y must agree in at least k bits. So they can disagree in at most $n-k$ bits, which means that $d(x,y) \leq n-k$. But $y \in A$ so $d(x,A) \leq n-k$, which completes the proof. ■

However, a low Hamming distance does not imply high similarity, as Anthony & Hammer showed (we state their result without proof):

Theorem 4.12 *Suppose* $A_k^s \neq \{0,1\}^n$. *Then* $\exists x \in \{0,1\}^n \backslash A_k^s$ *with* $d(x,A) = 1$.

4.6 Relationship between Hamming Distance and Convexification

The definition of convexification depends on the notion of Hamming distance, so one would intuitively expect a strong relationship. Suppose we have an element w and a set A, and recall that $d(w,A)$ is the Hamming distance from w to A, i.e. $d(w,A) = min_{y \in A} d(w,y)$. Then it is trivial to show that $d(w,A) = x \Rightarrow w \notin \mathbb{C}_j(A),\ \forall j < 2x$. However, we cannot deduce from the value of $d(w,A)$ that $w \in \mathbb{C}_k(A)$ for any k (simply consider a set A consisting of a single element).

4.7 Relationship between Similarity and Convexification

Similarity and convexity are quite different measures, although we can make some rather weak statements about the implications of a certain degree of similarity for the flanking distance and vice versa.

We first look at the implications of high similarity. Suppose we have a set $A \subset \{0,1\}^n$ and an element $w \notin A$. We show that we can bound the flanking distance if similarity is sufficiently high:

Theorem 4.13 *Suppose $s(w,A) = k$, and $k \geq \frac{n}{2}$. Define $j = n-k$. Then $w \in \mathbb{C}_{2j}(A)$.*

Proof. Without loss of generality suppose that $w = (11\ldots1)$; a similar argument works for any $w \in \{0,1\}^n$. Then since $s(w,A) = k$, $\exists x \in A$ with $x_\alpha = 1, \alpha = 1,\ldots,k$, and $\exists y \in A$ with $y_\beta = 1, \beta = n-k+1,\ldots,n$. Crucially, we show that $x \neq y$. Suppose $x = y$. Then $x_\alpha = 1,\ \alpha = 1,\ldots,k, n-k+1,\ldots,n$. But $k \geq \frac{n}{2} \Rightarrow n-k \leq k \Rightarrow n-k+1 \leq k+1 \Rightarrow x_\alpha = 1, \alpha = 1\ldots n \Rightarrow x = w \Rightarrow w \in A$. But $w \notin A$ by assumption. So our original assumption was false, and $x \neq y$.

Now $d(x,y) \leq 2(n-k) = 2j$ with equality iff $x_\alpha = 0, \alpha = k+1,\ldots,n$ and $y_\beta = 0, \beta = 1,\ldots,n-k$. In that case, $d(w,x) = j$ and $d(w,y) = j$, so $w \in \mathbb{C}_{2j}(A)$. Let $a = |\{x_\alpha : x_\alpha = 1, \alpha = k+1,\ldots,n\}|$ and let $b = |\{y_\beta : y_\beta = 1, \beta = 1,\ldots,n-k\}|$. Then $d(w,x) = j-a$ and $d(w,y) = j-b$, by definition. But $d(x,y) = 2j-a-b$ since changing some of the bits $x_\alpha (\alpha = k+1,\ldots,n)$ to 1 can only reduce $d(x,y)$ by the number of bits changed (since the corresponding bits of y are always 1, since $k \geq \frac{n}{2}$) and similarly changing some of the bits $y_\beta (\beta = 1,\ldots,n-k)$ to 1 can only reduce $d(x,y)$ by the number of bits changed (since the corresponding bits of x are always 1). So $d(x,y) = d(x,w) + d(w,y) = 2j-a-b \leq 2j$ which means that $w \in \mathbb{C}_{2j}(A)$ as required. ■

The bound from the previous theorem is tight; while it will sometimes be the case that $w \in \mathbb{C}_p(A)$ for some $p < 2j$, we cannot guarantee this.

Claim 4.14 *Given w and $k \geq \frac{n}{2}, k \neq n$, $\exists A \subseteq \{0,1\}^n$ such that $w \notin \mathbb{C}_p(A)$ for any $p < 2j$, where $j = n-k$ (i.e. $\mathbb{F}(w,A) \geq 2j$).*

Proof. Again without loss of generality suppose $w = (11\ldots1)$; a similar argument works for any element of $\{0,1\}^n$. Let A be the set of all points with exactly k bits equal to 1. Clearly $s(w,A) = k$. But

for any $a \in A$, $d(a,w) = j$. Suppose w is between $a_1, a_2 \in A$. Then $d(a_1,a_2) = d(a_1,w) + d(w,a_2) = 2j$. So $\mathbb{F}(w,A) \geq 2j$. ■

In the other direction, we can show that any degree of convexity implies 1-similarity.

Lemma 4.15 *Suppose $w \in \mathbb{C}_i(A)$ for some i. Then $s(w,A) \geq 1$*

Proof. By definition of convexity w is between some $x,y \in A$, i.e. $d(x,y) = d(x,w) + d(w,y)$. Now for each bit w_j, $j = 1,\ldots,n$, either $x_j \neq y_j$, in which case either $w_j = x_j$ or $w_j = y_j$, or $x_j = y_j$. Suppose that in that case $w_j \neq x_j$. Then define w' to be w except that $w'_j = 1 - w_j$. Now $d(w',x) = d(w,x) - 1$ since w' differs from x in one fewer bit than w does. Similarly $d(w',y) = d(w,y) - 1$. So $d(x,y) \leq d(x,w') + d(w',y) < d(x,w) + d(w,y) = d(x,y)$ which is a contradiction. So $w_j = x_j$. But then we have shown that for every bit w_j either $w_j = x_j$ or $w_j = y_j$. So $s(w,A) \geq 1$. ■

This bound is the best we can do, since:

Claim 4.16 *For any $i < n$ and $w \in \{0,1\}^n$, $\exists A \subseteq \{0,1\}^n$ such that $w \in \mathbb{C}_i(A)$ but $s(w,A) = 1$.*

Proof. Without loss of generality suppose that $w = (1\ldots.1)$. If i is even, i.e. $i = 2p, p \in \mathbb{Z}$ then define $x \in \{0,1\}^n$ s.t. $x_i = 1 \Leftrightarrow i \leq n - p$ and define $y \in \{0,1\}^n$ s.t. $y_j = 1 \Leftrightarrow j > p$. Then $d(x,y) = 2p = d(x,w) + d(y,w)$ (p is of course $\leq \frac{n}{2}$). If i is odd, i.e. $i = 2p+1, p \in \mathbb{Z}$, define $x \in \{0,1\}^n$ s.t. $x_i = 1 \Leftrightarrow i \leq n - p$ and define $y \in \{0,1\}^n$ s.t. $y_j = 1 \Leftrightarrow j > p+1$. Then $d(x,y) = 2p+1 = d(x,w) + d(y,w)$. Now let A be the set containing only the elements x,y. By construction, $w \in \mathbb{C}_i(A)$, but neither x nor y contains the positional substring of length 2 with bit 1 and bit n equal to 1, which is certainly in w. So $s(w,A) < 2$, but by the previous result we know that $s(w,A) \geq 1$. So $s(w,A) = 1$. ■

So it seems that the fact that w is in some convexification of A implies little about its similarity to A. This should not be surprising, as similarity is a strong requirement. For any given $k \in \mathbb{Z}^+$, w is a member of $\binom{n}{k}$ subcubes of $\{0,1\}^n$ of dimension $(n-k)$. The statement $s(w,A) \geq k$ implies that A contains some element of every one of those subcubes. An element of A can be in more than one such subcube, so A can have fewer than $\binom{n}{k}$ elements, but by contrast, the statement that $w \in \mathbb{C}_k(A)$ only requires that A contains 2 elements, say x,y, such that $d(x,y) = d(x,w) + d(w,y)$ and $d(x,y) = k$. The requirement imposed by Hamming distance is even less strict - $d(x,A) = k$ only requires A to contain one point y such that $d(x,y) = k$.

Could we strengthen the convexity requirement to try to find a stronger relationship with similarity? Under the existing definition, an element w is in $\mathbb{C}_i(A)$ if there is *at least one* pair $x,y \in A$ such that w is between x and y. Define the more restrictive set $\mathbb{C}_{ij}(A)$ to be all elements $w \in \{0,1\}^n$ that are between *at least j* pairs of elements of A each no more than distance i apart. These pairs must be distinct but do not have to be disjoint.

Definition 4.17 $\mathbb{C}_{ij}(A) = \{w \in \{0,1\}^n : \exists x_k, y_k \in A, k = 1,\ldots,j : d(x_k,w) + d(w,y_k) = d(x_k,y_k) \leq i\}$.

So by our previous definition of convexification, $\mathbb{C}_i(A) = \mathbb{C}_{i1}(A)$. This allows us to say slightly more about the relationship between convexity and similarity.

Lemma 4.18 $w \in \mathbb{C}_{22}(A) \Rightarrow s(w,A) \geq 2$.

Proof. $w \in \mathbb{C}_{22}(A)$, so at least 3 neighbours of w (i.e. elements of $\{0,1\}^n$ that differ from w in exactly one bit) are in A, say n_1, n_2, n_3. Suppose they differ from w in positions i, j, k respectively. Then any positional substring of w that is missing from n_1 must differ from n_1 in position i and similarly any positional substring of w that is missing from n_2 must differ from n_2 in position j, and any positional substring of w that is missing from n_3 must differ from n_3 in position k. So any positional substring of w that is missing from all elements of A must differ from them in positions i, j, k. But it must therefore be of length at least 3. ■

However, the following result shows that the measures are still quite different, even for low similarity.

Theorem 4.19 *Suppose* $w \in \{0,1\}^n$. *Let* $\alpha = 6n - 15$. *Then there is a set* A *such that* $w \in \mathbb{C}_{3\alpha}(A) \subseteq \{0,1\}^n$ *and* $|A| = 3n - 3$ *but* $s(w,A) = 2$.

Proof. Suppose without loss of generality that $w = (00\ldots0)$.

Now consider the set A which contains $x_1 = (1000\ldots0)$, $x_2 = (0100\ldots0)$ and $x_3 = (0010\ldots0)$. Suppose A also contains all $y_i \in \{0,1\}^n$ with either exactly 2 bits equal to 1 in positions 1,2,3, or with exactly 1 bit equal to 1 in positions 1,2,3 and 1 bit equal to 1 in positions 4,5,$\ldots,n$. Now there are $\binom{3}{2} + 3(n-3) = 3n-6$ vectors y_i, since there are $\binom{3}{2} = 3$ vectors with two 1's in positions $1,2,3$ and $3(n-3)$ with exactly one 1 in position 1,2 or 3 and one 1 in positions $4,\ldots,n$. So $\forall i, d(w,x_i) = 1$ and $\forall j, d(w,y_j) = 2$. Moreover, each x_j is distance 3 from exactly $1 + 2(n-3)$ of the y_i, and in that case $d(x_j,w) + d(w,y_i) = 1 + 2 = 3 = d(x_j,y_i)$, which means that w is between x_j and y_i. w is therefore between $3(1 + 2(n-3)) = 6n - 15$ pairs of elements of A, so $w \in \mathbb{C}_{3,(6n-15)}(A)$. But there is no element $a \in A$ such that a has 0 in positions 1,2,3. So the similarity of w to A is at most 2, as required. Further, since A contains the x_i (of which there are 3) and the y_j (of which there are $3n-6$), $|A| = 3 + 3n - 6 = 3n - 3$. ■

We conclude that a statement about convexity, or flanking, even with this modified definition is unlikely to guarantee a significant degree of similarity, and that the measures are quite different. In a sense, this is good news. Intuitively, there are many different ways that a new example (such as a new patient) can be "like" a set of previously seen examples (e.g. patients). Indeed if a new example

is very "like" previously seen examples in more than one way, intuitively we should be even more confident about classifying it based on experience of those examples than if it were very like the previously seen examples in one way but unlike them in another. Experimental results in Section 7 support this intuition; an example w which is both very similar to, and closely flanked by, a training sample A, is more likely to be accurately classified by a hypothesis h generated from A than a new example x which is, for example, very similar to A but not closely flanked by it.

5 Other Uses of Confidence Measures

Up to now, we assumed that we had a hypothesis h about the target function F and were using similarity, convexity, prevalence or Hamming distance to determine confidence. We can also use similarity and convexity for other purposes, including directly predicting $F(y)$ for a new example y.

5.1 Confidence Measures as a Means of Prediction

5.1.1 Similarity

In general, the fact that for some element $y \in \{0,1\}^n$ and training sample A we have $s(y,A) = k$ is not in itself enough to be sure about $F(y)$, even for large k. However, in certain circumstances we can be sure, as shown by the following result.

Lemma 5.1 *Suppose it is known that the target function $F \in D_{n,k}$ for some $n,k \in \mathbb{Z}$. Let $A \subseteq \{0,1\}^n$ be a set of examples, and split A into $A^- = \{x \in A : F(x) = 0\}$ and $A^+ = \{x \in A : F(x) = 1\}$. Then $s(y,A^-) \geq k \Rightarrow F(y) = 0$.*

Proof. Assume that $F(y) = 1$. Since $F \in D_{n,k}$ it can be written as a disjunction of monomials each of length $\leq k$. $F(y) = 1$ implies that at least one of these monomials of length $\leq k$ must give 1 on y. Suppose the monomial is dependent on the values of bits in positions $b_i, i = 1,\ldots,j, j \leq k$. But since $s(y,A^-) \geq k$ there must be $z \in A^-$ with the same values as y in the bits b_i. So the monomial must give the same result for z as for y, i.e. $F(z) = 1$. But this is a contradiction since $z \in A^-$. Hence our assumption that $F(y) = 1$ must be false, and the result follows. ■

Note that the proof does not rely on the hypothesis h. Unfortunately, the corresponding result with $s(y,A^+) \geq k$ does not hold, as the following counterexample shows.

Example 5.2 *Consider the function $F = u_3u_4 \vee u_2u_4 \vee u_2u_3 \vee u_1u_4 \vee u_1u_3 \vee u_1u_2$. Clearly $F \in D_{4,2}$, and $A^+ = \{(0011),(0101),(0110),(1001),(1010),(1100)\}$ is a set of positive examples. Consider $y = (0000)$. Now $F(z) = 1$, $\forall z \in A^+$, and all positional substrings of y of length 2 appear in at least one element of A^+, so $s(y,A^+) \geq 2$. But $F(y) = 0$.*

It is interesting that similarity makes it easier to determine whether a new example is a *negative* example, and suggests that we should consider the *complement* of F, $\overline{F}$, defined by $\overline{F}(x) = 1 - F(x),\ \forall x \in \{0,1\}^n$.

Corollary 5.3 *Let G be the complement of F. For any $y \in \{0,1\}^n$ with $s(y,C^+) \geq k$ for some C^+ a set of* positive *examples for G, $G(y) = 1$.*

The proof is immediate from the previous result, since C^+ is a set of negative examples for F.

But what is G? In general it will not be a $k-DNF$, but it is well known that if F has at most d terms in its DNF, its complement will have terms with at most d literals, i.e. $G \in D_{n,d}$. This leads to the following corollary:

Corollary 5.4 *Suppose it is known that the target function F can be represented by a DNF with at most d terms, i.e. $F \in D^d_{n,k}$ for some $k \in \mathbb{Z}^+$. Suppose A^+ is a set of positive examples of F. Then $\forall y \in \{0,1\}^n, s(y,A^+) \geq d \Rightarrow F(y) = 1$.*

Proof. Define $G(x) = 1 - F(x)$. We know that $G \in D_{n,d}$, and A^+ is a set of negative examples of G. So by previous lemma, $s(y,A^+) \geq d \Rightarrow G(y) = 0$. But $G(y) = 0 \Rightarrow F(y) = 1$. ■

We present experimental results in Section 7.3 which suggest that if a new example is more similar to the set of positive examples A^+ than the set of negative examples A^-, it is more likely to be a positive example.

5.1.2 Convexity

Ekin et al. (1998) defined a class of functions which they called *k-convex*.

Definition 5.5 *k-convex*
A function $f : \{0,1\}^n \rightarrow \{0,1\}$ is k-convex *if and only if for any two points $x,y \in \{0,1\}^n$ with $f(x) = f(y) = 1$ and $d(x,y) \leq k$, $f(z) = 1 \forall z\ s.t.\ d(x,y) = d(x,z) + d(z,y)$.*

In other words, a function f is k-convex if any point between two true points of f that are distance no more than k apart is itself a true point. To illustrate this in another way, if we consider the two points x and y as opposite corners of a subcube of co-dimension $\leq k$, all points in the subcube must also be true points. Alternatively, f is k-convex if and only if the k-convex hull of f is equal to f itself.

Clearly if we know that our target function F is k-convex for some $k \in \mathbb{Z}^+$, and we have a new example $x \in \{0,1\}^n$ which is flanked by the positive elements A^+ of the training sample, such that

$\mathbb{F}(x,A) \leq k$, we know that x is between two true points of the function distance no more than k apart and so it must be a true point, i.e. $F(x) = 1$. But how likely is it that our target function will be k-convex? Ekin et al. showed that if $\Phi(n,k)$ is the set of k-convex functions on $\{0,1\}^n$, we have:

$$2^{\frac{2^n}{\sum_{i=0}^{k}\binom{n}{i}}} \leq |\Phi(n,k)| \leq (3^k)^{2^{n-k}}$$

which even for relatively small n,k is a tiny fraction of the total number of possible Boolean functions, i.e. 2^{2^n}. However, we might suppose that while a randomly chosen Boolean function is unlikely to be k-convex, Boolean functions that are encountered in practice as the underlying function for some real-world phenomenon are generally not completely random and may have enough regularity to be k-convex, at least for small k.

5.2 Using Confidence Measures to Identify Outliers in a Sample

As well as asking how similar a new example x is to our training sample A we can ask, for each $a \in A$ how similar a is to $A\backslash a$, i.e. to the rest of the training sample. Our interest in this measure comes from the intuition that an example with low similarity to the rest of the sample is likely either to contain errors or to be a particularly unusual instance of a class which merits closer study on its own. We propose the following algorithm for determining similarity to the rest of the sample:

Algorithm 5.6 *j,k-Outliers*

Generate all monomials with $\leq j$ literals.
for each monomial m
    for each example $a \in A$
        If $m(a) = 1$ associate a with m. If m has $> k$ associated examples, discard it
    end for
end for
Let S be the set of outliers. Set $S = \emptyset$
for each remaining monomial m
    Add any associated examples to S.
end for
Output S

When this algorithm is run with $k = 1$, the output is all examples with similarity $\leq j$ to the rest of the sample A. The inclusion of the k parameter allows us to classify the members of a set T of up to k examples as outliers if the similarity of any $t \in T$ to $A\backslash T$ is $\leq j$.

The similarity of $a \in A$ to $A\backslash a$ is closely related to the *specification number* of a, as defined by Anthony, Brightwell & Shawe-Taylor (1995).

Definition 5.7 *Specification Number*
The specification number of $a \in A$ is the smallest number of bits of a which we need to know to distinguish a from any other element of A.

Another way of writing this definition is that the specification number is the length of the shortest positional substring that distinguishes a from the rest of A. But this is just 1 more than $s(a, A \backslash a)$.

How long does the algorithm take? Recall that computing exactly the similarity of an arbitrary $w \in \{0,1\}^n$ to a set $B \subseteq \{0,1\}^n$ is NP-hard, and so we cannot expect to compute the specification number in polynomial time. However, the algorithm given here computes, for each $a \in A$ whether $s(a, A \backslash a) \leq j$ for fixed j. As we have seen before (Algorithm 4.9 on page 15), this can be done efficiently. The first two *for* loops compare each monomial with each example at most once, which will take $O(|A|n^j)$ time. The last *for* loop takes $O(kn^j)$ time, since each monomial is examined at most once, and has at most k associated examples. So the algorithm runs in $O((|A|+k)n^j)$ time, which is of the same order of magnitude as learning a $D_{n,k}$ function (Algorithm **??** on page **??**).

As well as similarity to the whole sample, we can split A into $A^- \cup A^+$ where A^- is the set of all negative examples, and A^+ the set of all positive examples. We then compute outliers for A^+ and A^- separately, remove them, and recalculate our hypotheses. We propose the term *removing type outliers* for this process, and the term *removing sample outliers* if we compute outliers for A as a whole and then remove them. Note that the process of removing type outliers will always remove at least as many elements as removing sample outliers, since the similarity of a positive (negative) element to A^+ (respectively A^-) cannot be higher than the similarity of that element to A as a whole, and could well be lower.

We ran experimental tests on standard datasets (listed in Section 7), identifying possible outliers using similarity and removing them. However any improvement in accuracy was limited, or we had to classify a significant fraction of the dataset as outliers to see any improvement. The most promising experiment took the difficult Pima Indian Diabetes dataset (see Section 7.1.2) and removed any *type outliers* that had a similarity of less than 4 to the set of positive or negative examples in the sample. There were 166 such, out of 392 examples. We show later that this dataset is inconsistent with an underlying function in $D_{16,5}$. However, with the type outliers removed, the dataset was found to be consistent with $D_{16,5}$, and indeed with the simpler hypothesis space $D_{16,4}$. Classification accuracy also increased to 82.9% with type outliers removed, against 65.6% for the dataset as a whole (although given the number of examples that we had to remove, perhaps this is not surprising). We suggest some avenues for further work in Section 8.1.5.

6 Modified $D_{n,k}$ Learning Algorithm including Confidence Measure

The standard $D_{n,k}$ learning algorithm (Algorithm **??** on page **??**) is efficient, but the hypotheses it generates tend to classify too many examples as positive examples. Valiant recognised this and termed it a *no false negatives* algorithm, i.e. the hypotheses it produced would never wrongly classify a new example as negative when it was actually a positive example. The following lemma shows why.

Lemma 6.1 *Let h be the hypothesis produced by the standard $D_{n,k}$ learning algorithm, with $F \in D_{n,k}$ the underlying Boolean function, and let y be a new example, with $h(y) = 0$. Then y is a negative example.*

Proof. We show that $s(y,N) \geq k$ where N is the set of previously seen negative examples, and hence by Lemma 5.1 it must be a negative example. Suppose this is false. Then there is a positional substring of y, of length $\leq k$, which does not appear in any $x \in N$. Consider the corresponding monomial m. This is not satisfied by any $x \in N$, and so will not have been removed from h by the standard DNF learning algorithm. So $h(y) = 1$, which is a contradiction, and the result follows. ■

So the hypothesis produced by the standard $D_{n,k}$ learning algorithm classifies a new example as 1 unless it absolutely has to classify it as 0 for consistency, which implies that the hypothesis produced by the standard $D_{n,k}$ learning algorithm will sometimes classify an example as positive when it would be better to classify it as negative, or simply to say "don't know". We present a new algorithm which attempts to produce a more reasonable hypothesis.

Algorithm 6.2 *$D_{n,k}$ Learning Algorithm including Confidence Measure*

Suppose A is of size s and the examples are of length n.
Suppose we know that the underlying function F is in $D_{n,k}$ for some k
Generate functions d_i where each one is the disjunction of all monomials of length exactly i, for $i = 1,\ldots,k$
Set the initial hypothesis h to be the identically 0 function.
for each $y \in A$
  if y is a positive example then
    for each monomial m in h
      if $m(y) = 1$ add 1 to the prevalence count for m.
    end for
    for $j = 1,\ldots,k$
      for each monomial m in d_j
        if $m(y) = 1$ then move m to h and set the prevalence count of m to 1.
      end for

end for
else (y must be a negative example)
for each monomial m in h
if $m(y) = 1$ *remove m from h*
end for
for $j = 1, \ldots, k$
for each monomial m in d_j
if $m(y) = 1$ *remove m from* d_j
end for
end for
end if
end for
output h as our hypothesis.

Lemma 6.3 *The hypothesis h is consistent*

Proof. We proceed by induction. h is consistent after 0 iterations of the main *for* loop. Suppose h is consistent after $n \geq 0$ iterations. At iteration $n+1$ we either:
a) process a negative example. We remove all monomials in h and the d_i that give 1 on the negative example. h and the d_i are clearly still consistent on previously seen negative examples since removing monomials from a DNF cannot change the classification given to an example from type 0 to type 1. For any previously seen positive example p, h will contain the set M of all monomials of length $\leq k$ that are true on that positive example and on no negative examples. h will classify p as a positive example unless all elements of M are removed. But then $s(p, A^-) \geq k$, which by Lemma 5.1 means that p must be a negative example, which is a contradiction. So $h(p) = 1$ as desired.
b) process a positive example. We move any monomials in the d_i that satisfy the example to h. h is clearly still consistent on all previously seen positive examples, since adding monomials to a DNF cannot change the classification given to an example from type 1 to type 0. Suppose h gives 1 for a previously seen negative example. Any monomials that were in h before processing this positive example give 0 on all previous negative examples, by inductive assumption. So one of the monomials that we moved to h at this step must give 1 on a previous negative example. But we know that the d_i give 0 on all previously seen negative examples, by construction. This contradiction shows that h must be consistent on previously seen negative examples, which completes the proof. ■

So h is consistent, which means that in the PAC framework, h will be a probably approximately correct hypothesis, given a large enough training sample. It is also *memoryless online*. Further, we can recover the hypothesis that would have been generated by the standard $D_{n,k}$ learning algorithm; it is just the disjunction of h with all the d_i.

The running time of the algorithm is $O(sn^k)$, since the main loop repeats $s = |A|$ times, and each iteration through the loop involves evaluating all the remaining monomials once. To see that each iteration takes $O(n^k)$ time, recall that the number of monomials of length at most k is bounded above by $\sum_{j=1,\ldots,k} \binom{2n}{j}$, which is less than $(2n)^k = O(n^k)$. This is the same running time as the standard $D_{n,k}$ learning algorithm.

However, this algorithm gives us more than a potentially PAC hypothesis. Suppose we have a hypothesis h and a new example z. There are 3 possible cases:

1. $h(z) = 1$. In this case some of the positional substrings in z have only previously been seen in positive examples, which suggests that z is also a positive example. But if we are cautious, we may wish the algorithm to output "don't know" if the prevalence count $p(z,h,A)$ is low, or even to classify z as a negative example. Experimental evidence in Section 7 suggests that when p is less than 10 or about 5 percent of the sample, we should certainly classify z as "don't know" at best, and indeed the hypothesis is often improved if we remove low prevalence monomials from it altogether (which would classify z as type 0). However, the hypothesis would no longer necessarily be consistent.

2. $h(z) = 0$ and $d_i = 0 \; \forall i$. Let h_d be the disjunction of h with all the d_i (i.e. the hypothesis generated by the standard $D_{n,k}$ learning algorithm). Then $h_d(z) = 0$. Lemma 5.1 implies that z must be a negative example.

3. $h(z) = 0$ and $d_i = 1$ for some i. Consider any monomial in d_i that is satisfied by z. It cannot correspond to a positional substring from any negative example, as that would have been removed by the algorithm. Similarly it cannot correspond to a positional substring appearing only in a positive example, as it would then be in h (which would mean that $h(z) = 1$). So it must represent a positional substring which does not appear in *any* member of the training sample. To put it another way, z is at most $(i-1)$-similar to the training sample. When i is small, we should say "don't know" as we have no evidence either way. For i large, we might feel justified in supposing that z is a negative example.

So we can post-process the classification given by the hypothesis to any new example by applying two *confidence parameters*: a minimum prevalence count, below which a new example cannot be classified as type 1, and a minimum similarity, below which a new example cannot be classified as type 0. The exact values of these confidence parameters will vary depending on the trade off we are willing to make between not classifying an example at all (i.e. "don't know") and the risk that our classification will be wrong. We consider the use of prevalence count in Section 7.

6.1 An Example Using the Algorithm

Consider a simplified medical diagnosis agent. This agent will be designed to diagnose a single disease (call it 'bronchitis') and will only consider 4 symptoms:
1. Cough
2. Fever
3. Headache
4. Positive Chest X-Ray

We suppose that the actual diagnostic rule for bronchitis is: Cough AND Fever AND NOT(Positive Chest X-Ray); for the sake of this example assume that a Positive Chest X-Ray indicates some more serious disease. Whether or not the patient has a Headache is irrelevant to the diagnosis.

Suppose the agent uses the algorithm above. How does its hypothesis change during the learning process, as it sees new examples (i.e. patients)?

The first patient has *Cough AND Fever AND Positive X-Ray AND NOT (Headache)*. The agent is told that he does not have bronchitis. It does not modify its basic hypothesis h, which remains at the identically 0 function, but does remove those monomials that give 1 on this patient from the d_i functions. So for example d_1 is now *Headache OR NOT (Cough) OR NOT (Fever) OR NOT (Positive X-Ray)*, reflecting the fact that the agent has not yet seen any patients with these symptoms.

The second patient has *Cough AND Fever AND Headache AND NOT (Positive X-Ray)*. She does have bronchitis. The agent adds all possible combinations of symptoms from this patient to its hypothesis. The hypothesis is now:
(Cough AND Fever AND Headache)
OR (Cough AND Fever AND NOT Positive X-Ray)
OR (Cough AND Fever AND Headache AND NOT Positive X-Ray)
OR (Cough AND Headache and NOT Positive X-Ray)
OR (Fever AND Headache and NOT Positive X-Ray)
OR (Headache AND NOT Positive X-Ray)
OR (Cough AND Headache)
OR (Cough AND NOT Positive X-Ray)
OR (Fever AND Headache)
OR (Fever AND NOT Positive X-Ray)
OR Headache
OR (NOT Positive X-Ray).

Note that it does not include, for example, the set of symptoms *Cough OR Fever OR (Cough AND Fever)*, since this set was excluded by the previous negative example.

The third patient has *Cough AND Headache AND NOT(Fever) AND NOT(Positive X-Ray)*. He

does not have bronchitis. The agent removes all monomials in its hypothesis that give 1 on this patient, and does the same for the d_i functions. The hypothesis is now:
(Cough AND Fever AND Headache)
OR (Cough AND Fever AND NOT Positive X-Ray)
OR (Cough AND Fever AND Headache AND NOT Positive X-Ray)
OR (Fever AND Headache and NOT Positive X-Ray)
OR (Fever AND NOT Positive X-Ray)
OR (Fever AND Headache).

The fourth patient has a *Cough AND NOT(Fever) AND NOT(Headache) AND NOT(Positive X-Ray)*. He does not have bronchitis. The hypothesis is unchanged, but any monomials in the d_i functions that give 1 on this patient would be removed.

We now consider that our agent is fully trained, and present it with two new patients for it to classify.

The first new patient has *Fever AND Positive X-Ray AND NOT(Headache) AND NOT(Cough)*. The hypothesis would correctly classify this patient as not having bronchitis. If the agent post-processes the hypothesis by refusing to diagnose any patient with a previously unseen symptom (equivalent to refusing to classify any new example with similarity 0 to the training sample), it would reply "don't know", since it has never seen a patient without a Cough. Note that the hypothesis from the standard $D_{n,k}$ learning algorithm would diagnose bronchitis in this patient.

The second new patient has *Cough AND Fever AND NOT(Headache) AND NOT (Positive X-Ray)*. The agent correctly classifies this patient as having bronchitis, provided that it is not concerned with the prevalence count. If the agent post-processes the hypothesis by refusing to diagnose based on a set of symptoms until it has seen that set of symptoms in at least $N > 1$ patients in the training sample, it would say "don't know" for this patient (and indeed would be unwilling to classify *any* patient as having bronchitis until we have given it some more training examples that it is told have bronchitis). We will see in Section 7 that refusing to classify based on low prevalence monomials is often the correct strategy, even if it is not in this particular case.

7 Experimental Results with Confidence Measures

We now present the results of a number of experiments using the confidence measures and algorithms presented in this paper. The algorithms were implemented in $C^\sharp$, a language similar to Java and C++, using the free open source SharpDevelop environment (Kruger et al. (2000)). Veal (2005b) has used this code to develop a learning application, and full source code for that application, which includes the algorithms presented in this paper, can be found on his website.

The experiments use the following standard datasets from the Machine Learning Database at UCI

(Hettich et al. 1998):

1. A dataset based on a study of breast cancer patients at the University of Wisconsin. This dataset (which we refer to as Breast Cancer Wisconsin, or BCW) is known to be relatively easy to produce good hypotheses for. Patients in the study are classified into two types, corresponding to the type of their disease: benign or malignant. The hypothesis generated by our algorithm attempts to classify them based on their symptoms and the results of various medical tests.

2. A dataset based on a study of the incidence of diabetes among the Pima Indians. This dataset (which we refer to as Pima Indian Diabetes, or PID) is known to be difficult to produce good hypotheses for. Patients are classified as either having diabetes or not. As for the BCW dataset, the hypothesis generated by our algorithm attempts to classify the patients based on their symptoms and the results of various medical tests.

3. A dataset based on the study of radar reflections from the ionosphere. We include this dataset (which we refer to as Ionosphere) to broaden our experimental data beyond medical datasets, which similarity might be expected to be particularly useful for. The dataset includes two types of radar observations of particles in the ionosphere, classified as "good" (there was radar reflection) or "bad".

4. A dataset based on voting records from the US Congress. We include this dataset (which we refer to as Voting) for two reasons: to further reduce any bias towards medical datasets, and because this dataset is already binarized (voting records are naturally binary since a voter is either *for* or *against* a proposal). This means that we can avoid the binarization step and assess the learning algorithm in isolation. Each element in this dataset represents a single Congressman/ Congresswoman. The aim is to determine the party that each one belongs to, based on their voting record. This dataset is known to be easy to produce good hypotheses for.

We first removed any examples in the datasets which had missing input values (i.e. the value of some input was not given for that example). We then transformed the numeric inputs into binary ones using the process outlined in Section 3 (except for the Voting dataset), then took each dataset and partitioned it at random into four subsets. We removed one subset (call it the test set) and used the other three subsets as a training sample for our learning algorithm (Algorithm 6.2 on page 24). We then applied the generated hypothesis to the test set and for each example in that subset, compared the classification given by the hypothesis with the true classification. We repeated this process for each of the four subsets. At the end of this process, which is known as *4-fold cross validation*, each item x in the dataset had been in the test set exactly once, and we had a classification for it based on a hypothesis generated by our algorithm. Clearly this process could be generalised to p-fold cross

validation by partitioning into p subsets. We repeated this process 10 times, with a different random partitioning each time, and took the average accuracy of the hypotheses generated.

7.1 Does High Confidence Imply High Accuracy?

We looked at whether the fact that an example is "like" the training sample, in the sense of any of the confidence measures given above, implied that our hypothesis was more likely to be correct for that example than for other examples which were less "like" the training sample. To do this, we calculated, for each example y in each test set:

1. the similarity to the corresponding training sample A i.e. $s(y,A)$;
2. the degree to which the item is flanked by A, i.e. $\mathbb{F}(y,A)$;
3. the Hamming distance from y to A i.e. $d(y,A)$;
4. for an example classified as type 1 by the algorithm, the maximum prevalence count of any monomials that the example satisfies, i.e. $p(y,h,A)$.

We then group the items by similarity, flanking distance, Hamming distance and prevalence, and compute the accuracy of the hypothesis for each group. Accuracy is defined as the percentage of items in each group correctly classified by our hypothesis. For example, we look at the accuracy on all items y such that $s(y,A) \geq 5$ and the accuracy for those items such that $\mathbb{F}(y,A) \leq 3$. We also do pairwise comparisons; for example, we look at all items y such that both $d(y,A) \leq 2$ *and* $s(y,A) \geq 5$.

In the tables of results in this section, the numbers in brackets show the number of examples in each group. The percentage figures are the accuracy of the generated hypotheses on examples in that group.

7.1.1 Wisconsin Breast Cancer Dataset

We start with probably the easiest dataset, the University of Wisconsin Breast Cancer study. We manipulate it as above, and find a support set with 12 binary inputs. We look for a hypothesis in $D_{12,5}$. Recall that the output of the algorithm includes the similarity functions (called d_i) for similarity $i \leq 5$. Table 1 shows the accuracy on the test set as a whole, and accuracy on subsets of the test set with flanking distance and/ or similarity constrained (note that the category ALL includes all items within the dataset, even items that are not flanked by A, or have low similarity to A). The improvement in accuracy for high similarity and close flanking is strikingly visible when shown graphically (see Figure 2).

Flanking/ Similarity	ALL	$s(y,A) \geq 4$	$s(y,A) \geq 5$	$s(y,A) \geq 6$
ALL	(680) 94.9%	(670) 95.4%	(620) 97.6%	(584) 99.3%
$\mathbb{F}(y,A) \leq 5$	(679) 94.9%	(670) 95.4%	(620) 97.6%	(584) 99.3%
$\mathbb{F}(y,A) \leq 4$	(674) 95.1%	(667) 95.5%	(619) 97.6%	(584) 99.3%
$\mathbb{F}(y,A) \leq 3$	(656) 95.4%	(652) 95.7%	(615) 97.7%	(581) 99.3%
$\mathbb{F}(y,A) \leq 2$	(624) 97.1%	(621) 97.2%	(604) 98.3%	(576) 99.5%

Table 1: Wisconsin Breast Cancer Data: Accuracy vs Similarity and Flanking Distance

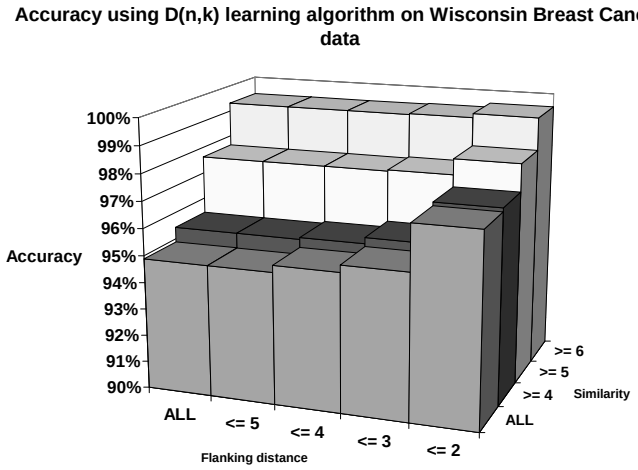


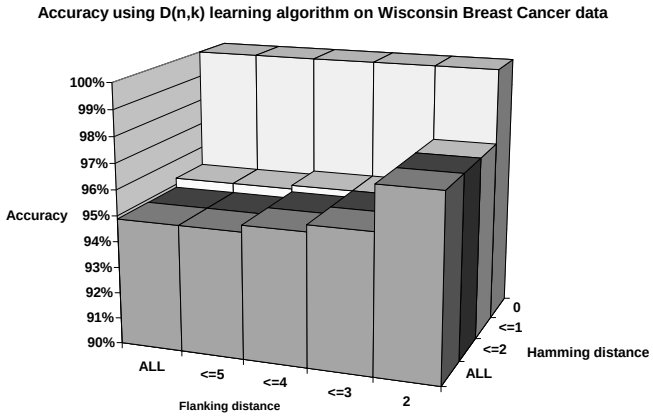
Figure 2: Wisconsin Breast Cancer Data: Accuracy vs. Similarity and Flanking Distance

Flanking/ Hamming distance	ALL	$d(y,A) \leq 2$	$d(y,A) \leq 1$	$d(y,A) = 0$
ALL	(680) 94.9%	(679) 94.9%	(662) 95.3%	(566) 100.0%
$\mathbb{F}(y,A) \leq 5$	(679) 94.9%	(679) 95.3%	(662) 95.3%	(566) 100.0%
$\mathbb{F}(y,A) \leq 4$	(674) 95.1%	(674) 95.4%	(660) 95.4%	(566) 100.0%
$\mathbb{F}(y,A) \leq 3$	(656) 95.4%	(656) 95.4%	(656) 95.4%	(563) 100.0%
$\mathbb{F}(y,A) \leq 2$	(624) 97.1%	(624) 97.1%	(624) 97.1%	(560) 100.0%

Table 2: Wisconsin Breast Cancer Data: Accuracy vs Hamming Distance and Flanking Distance

We can also look at how accuracy changes when we impose restrictions on the Hamming distance of the new examples from the training sample, and on the flanking distance. This is shown in Table 2 on page 32 and Figure 3.

Figure 3: Wisconsin Breast Cancer Data: Accuracy vs. Hamming Distance and Flanking Distance



Interestingly, the hypothesis is 100% accurate on any new example which is at Hamming distance 0 from the training sample (i.e. identical to an example in the training sample). In other words, the hypothesis is *consistent*, which implies that the BCW data is consistent with a hypothesis in $D_{12,5}$. We will see later that this is not the case for the more difficult PID dataset.

Finally, we consider the usefulness of the *prevalence count*. Recall that each monomial m_i in the output hypothesis h is true on a certain number of positive examples in the training sample (and no negative examples). A new example x which is classified as a positive example must satisfy at least one of these monomials. Take the monomial with the highest prevalence count that is satisfied, and as in Section 4.2, call this the *maximum positive prevalence* of x on A, $p(x,h,A)$. Now graph the accuracy of our hypothesis h on x against the maximum positive prevalence of x. Figure 4 on page 33 shows that, as expected, classification based on a low prevalence count is unreliable (in the figure, the column labelled q covers all values of maximum positive prevalence between q and $q+9$). We note that for a maximum positive prevalence below 50, the hypothesis is no better than chance, which

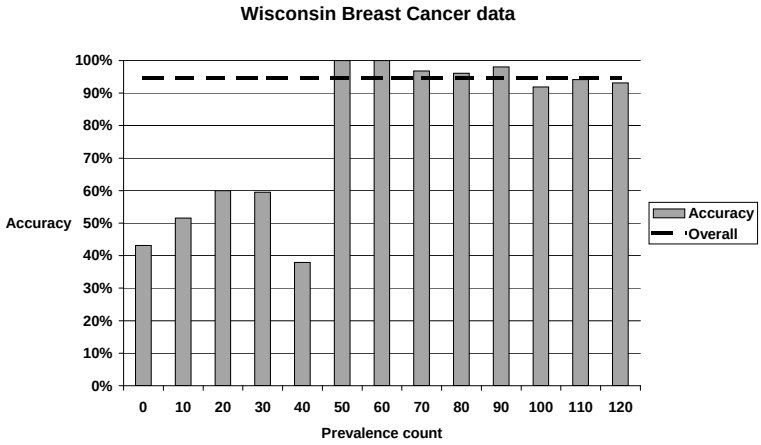


Figure 4: Wisconsin Breast Cancer Data: Accuracy vs. Prevalence Count

suggests that "don't know" would be an appropriate response from it. Above that level, increasing prevalence appears to make little difference to the accuracy.

7.1.2 Pima Indian Diabetes Dataset

Now consider the Pima Indian Diabetes (PID) dataset. When binarized, the dataset has 16 inputs per example, and we look for a hypothesis in $D_{16,5}$. The accuracy of this hypothesis for given similarity and flanking distance is shown in Table 3 on page 34 and in Figure 5 on page 35. We also show the accuracy of the hypotheses for given similarity and Hamming distance in Table 4 on page 34 and Figure 6 on page 35. Note that the accuracy on new examples which are at (Hamming) distance 0 from the training sample (i.e. identical to examples in the training sample) is 98.6%. Recall that our learning algorithm is consistent, and so (Definition 2.8 on page 5) given a consistent training sample it will produce a consistent hypothesis. Since the hypothesis here is *not* consistent, the PID data must be inconsistent with a hypothesis in $D_{16,5}$ (compare the BCW data, which appears to be consistent with a hypothesis in $D_{12,5}$). We also note that the hypothesis contains a number of monomials with low prevalence, which means that the hypothesis tends to classify too many examples as type 1. Recall that in our learning algorithm we introduced a confidence parameter which related to the prevalence count. If we post-process the algorithm output using this confidence parameter by assuming that any classification as type 1 based on a monomial with prevalence less than 5 is incorrect, and classify the example as 0 instead, overall accuracy jumps to 75.2% (vs. 66.1% "uncorrected"). This compares well with the best classification accuracy on the PID data by known learning algorithms. Note that all examples are still classified; we could if we were more cautious assume that any classification based on a monomial with prevalence less than 5 should be changed to "don't know". However, in considering prevalence count, we find that there is no monomial which is true on more than 20 of the

Flanking/ Similarity	ALL	$s(y,A) \geq 3$	$s(y,A) \geq 4$	$s(y,A) \geq 5$	$s(y,A) \geq 6$
ALL	(392) 66.1%	(392) 66.0%	(373) 66.4%	(292) 72.3%	(182) 84.0%
$\mathbb{F}(y,A) \leq 6$	(392) 66.1%	(391) 66.0%	(373) 66.4%	(292) 72.3%	(182) 84.0%
$\mathbb{F}(y,A) \leq 5$	(386) 66.5%	(386) 66.5%	(370) 66.4%	(292) 72.3%	(182) 84.0%
$\mathbb{F}(y,A) \leq 4$	(369) 67.2%	(369) 67.1%	(358) 67.0%	(289) 72.5%	(181) 84.0%
$\mathbb{F}(y,A) \leq 3$	(320) 70.2%	(320) 70.1%	(312) 70.0%	(264) 74.1%	(174) 84.3%
$\mathbb{F}(y,A) \leq 2$	(233) 74.2%	(233) 74.2%	(231) 74.4%	(210) 77.1%	(157) 84.5%

Table 3: Pima Indian Diabetes Data: Accuracy vs Similarity and Flanking Distance

Hamming/ Similarity	ALL	$s(y,A) \geq 3$	$s(y,A) \geq 4$	$s(y,A) \geq 5$	$s(y,A) \geq 6$
ALL	(392) 66.1%	(392) 66.0%	(373) 66.4%	(292) 72.3%	(182) 84.0%
$d(y,A) \leq 2$	(387) 66.5%	(387) 66.5%	(371) 66.5%	(292) 72.3%	(182) 84.0%
$d(y,A) \leq 1$	(329) 70.6%	(329) 70.6%	(320) 70.5%	(270) 74.6%	(179) 84.8%
$d(y,A) = 0$	(102) 98.6%	(102) 98.6%	(102) 98.6%	(102) 98.6%	(102) 98.6%

Table 4: Pima Indian Diabetes Data: Accuracy vs Similarity and Hamming Distance

130 positive examples in the dataset (about 15%), so we cannot say that any one example classified as a positive example by the hypothesis is supported by a monomial of substantially higher prevalence than the other. The situation was quite different for the BCW dataset; the hypothesis for that dataset contained monomials which were true on over half the 239 positive examples (and by assumption, none of the negative examples). These high prevalence count monomials presumably represent broad underlying structure in the data, which our algorithm has not found in the PID dataset. This agrees well with the work in Hammer & Bonates (2005), where the maximum prevalence (as opposed to prevalence count) of a monomial true only on positive examples was 54.2% for the BCW dataset, but only 16.3% for the PID dataset (they used a slightly more sophisticated binarization algorithm, so the results may not agree exactly).

7.1.3 Ionosphere Dataset

We now turn to non-medical data, specifically the Ionosphere data mentioned above. This data relates to radar investigation of particles in the ionosphere; the results of each investigation are classified in the dataset as "good" or "bad".

We ran the same experiments using this dataset as for the BCW and PID datasets, and the results are shown below. Figure 7 on page 35 and Table 5 on page 36 show the accuracy of the hypotheses against similarity and flanking distance. Note that restricting the hypotheses to those examples that are closely flanked and very similar to the training sample gives high accuracy, but as for the other datasets we cannot say that in all cases $s(x,A) > s(y,A)$ means that we should be more confident about the classification of example x than that of y.

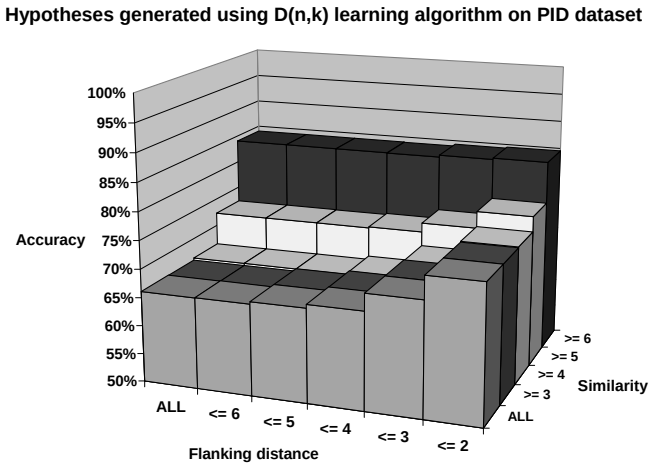


Figure 5: Pima Indian Diabetes Data: Accuracy vs. Similarity and Flanking Distance

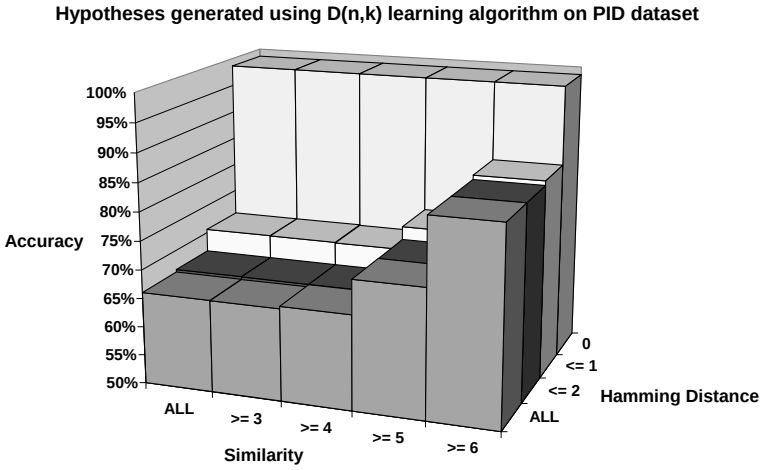


Figure 6: Pima Indian Diabetes Data: Accuracy vs. Similarity and Hamming distance

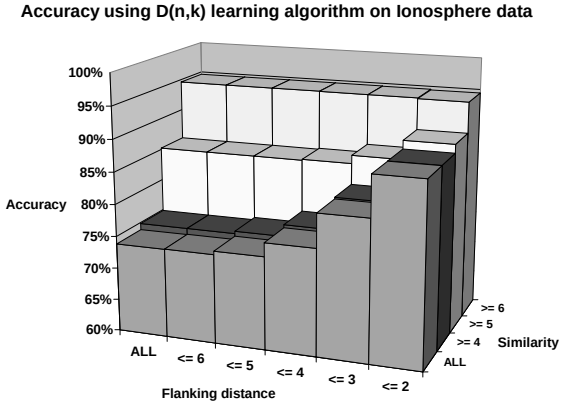


Figure 7: Ionosphere Data: Accuracy vs. Similarity and Flanking Distance

Flanking/ Similarity	ALL	$s(y,A) \geq 4$	$s(y,A) \geq 5$	$s(y,A) \geq 6$
ALL	(348) 73.9%	(343) 74.8%	(282) 85.2%	(209) 94.6%
$\mathbb{F}(y,A) \leq 6$	(348) 73.9%	(343) 74.8%	(282) 85.2%	(209) 94.6%
$\mathbb{F}(y,A) \leq 5$	(345) 74.4%	(340) 75.2%	(282) 85.2%	(209) 94.6%
$\mathbb{F}(y,A) \leq 4$	(331) 76.6%	(329) 77.1%	(280) 85.3%	(208) 94.6%
$\mathbb{F}(y,A) \leq 3$	(299) 81.8%	(299) 82.1%	(269) 86.6%	(207) 94.5%
$\mathbb{F}(y,A) \leq 2$	(241) 88.2%	(241) 88.2%	(233) 89.6%	(194) 94.4%

Table 5: Ionosphere Data: Accuracy vs Similarity and Flanking Distance

Interestingly, when we consider the accuracy of the hypothesis against prevalence count (as we did for the BCW dataset earlier - see Figure 4 on page 33) a similar picture emerges, in that for a prevalence count above 50, the hypothesis is very accurate; see Figure 8 on page 36. Note that in both cases we are looking for a hypothesis in $D_{12,5}$. This merits further investigation to see if some theoretical or empirical rule can be derived which would give a bound on the minimum prevalence count to use for classification, given the size of the dataset, the size of the example space (i.e. the value of n in $\{0,1\}^n$) and the maximum number of literals in any monomial in the hypothesis.

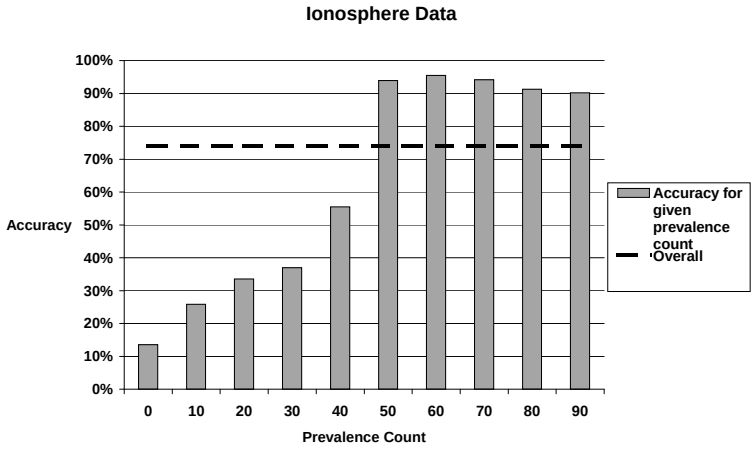


Figure 8: Ionosphere Data: Accuracy vs. Prevalence Count

7.1.4 Voting Records Dataset

Finally, consider the Voting Records dataset. We ran the same experiments using this dataset as for the BCW, PID and Ionosphere datasets, with the exception that there was no need to binarize the data. There are 16 binary inputs, and we look for a hypothesis in $D_{16,5}$, as for the PID data. The results are shown in Table 6 on page 37 and Figure 9 on page 37. Increasing similarity and close flanking are good measures of confidence, but require us to only classify a small part of the dataset.

Flanking/ Similarity	ALL	$s(y,A) \geq 3$	$s(y,A) \geq 4$	$s(y,A) \geq 5$	$s(y,A) \geq 6$
ALL	(232) 70.9%	(228) 71.4%	(189) 77.1%	(128) 91.1%	(100) 99.0%
$\mathbb{F}(y,A) \leq 6$	(213) 74.8%	(211) 75.0%	(185) 78.3%	(127) 91.1%	(100) 99.0%
$\mathbb{F}(y,A) \leq 5$	(194) 78.8%	(192) 79.0%	(176 80.7%	(127) 91.1%	(100) 99.0%
$\mathbb{F}(y,A) \leq 4$	(176) 83.1%	(175) 83.0%	(166) 84.0%	(126) 92.0%	(100) 99.0%
$\mathbb{F}(y,A) \leq 3$	(154) 88.6%	(153) 88.5%	(148) 88.6%	(123) 93.3%	(100) 99.0%
$\mathbb{F}(y,A) \leq 2$	(125) 94.3%	(125) 94.3%	(123) 94.2%	(115) 94.9%	(98) 99.3%

Table 6: Voting Data: Accuracy vs Similarity and Flanking Distance

The overall classification accuracy is however disappointing, as other algorithms are known to be able to classify this dataset with well over 90% accuracy. This appears to be at least in part due to the large number of inputs, some of which are unnecessary. We can find, using part of our binarization algorithm, a support set of size 8. If we then re-analyse the data, the accuracy does improve to over 90%. Again, prevalence count appears to play an important role; the highest prevalence monomial in the hypothesis gives a classification about as accurate as the hypothesis as a whole. Appendix B shows that the monomial makes good intuitive sense as a rule when translated into what it actually represents in the data.

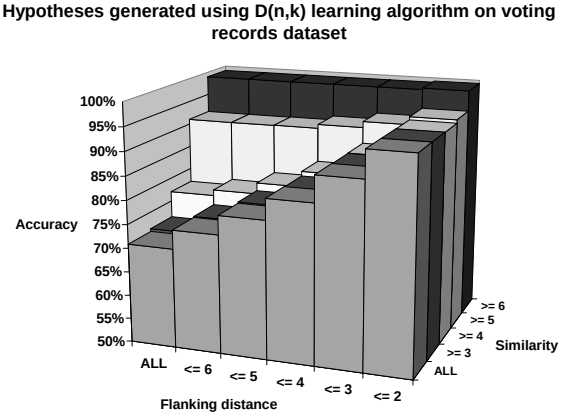


Figure 9: Voting Data: Accuracy vs. Similarity and Flanking Distance

7.2 Comparison of Different Convexification Hierarchies

In Section 4.3, we described a number of different convexification hierarchies. A full analysis of the hierarchies is a paper in itself, and beyond the scope of this paper, but we present here some experimental results on the single application convexification hierarchy in particular. We use two of the standard datasets previously mentioned, the Wisconsin Breast Cancer dataset and the Ionosphere dataset, and investigate the number of elements (or points) of $\{0,1\}^n$ found by applying a single *k*-

convexification to the base training sample A, for $k = 2,3,\ldots,n$ (and at that number as a fraction of the size of all possible examples, i.e. of $\{0,1\}^n$). For both BCW data and the Ionosphere data, $n = 12$, which means that there are $2^{12} = 4096$ possible points. Each slice of the pie chart in Figure 10 on page 39 represents the number of points that are between two elements $x,y \in A$ where $d(x,y) = k$ and are not between any two elements of A distance less than k apart. The labels on the slices give the number of points in the slice. For example, the slice "6 convex, 737" shows that there are 737 points in $\{0,1\}^n$ that are between two elements $x,y \in A$ where $d(x,y) = 6$ and are not between any two elements of A distance less than 6 apart.

Note that since k-convexification adds all points between elements of A distance *at most* k apart, a k-convexification would add all the elements represented by the slices $2,3,\ldots,k$. Also, that all points in $\{0,1\}^{12}$ are between *some* pair of elements of A. Finally, the size of the base sample (the number of elements in A) is given here as 154, but in earlier sections the value 680 was given. The difference is due to the fact that for many of the examples in the BCW dataset, there are several others with the same binary inputs, and which therefore correspond to the same element of $\{0,1\}^{12}$. The figure of 154 represents the number of distinct elements of $\{0,1\}^{12}$ in the dataset.

Now consider the ionosphere dataset. We are again working in $\{0,1\}^{12}$, and the convexification hierarchy looks not dissimilar, as shown in Figure 11 on page 39; again all the points in $\{0,1\}^{12}$ are flanked by elements of the dataset.

We also ran some experimental tests on the *iterated convexification* hierarchy, but as noted above, it turns out to be uninteresting for BCW (and indeed for PID and the Ionosphere data as well). The set of examples that is the result of applying iterated 2-convexification to the datasets is either the whole of $\{0,1\}^n$ or very nearly so, and $\mathbb{C}_i^*(A) = \{0,1\}^n, \forall i \geq 3$. This makes it impossible to distinguish examples on the basis of where they fit in the $\mathbb{C}_i^*(A)$ hierarchy. Simple experiments with the *increasing convexification* hierarchy suggests that it is less discriminating than single application convexification, but more so than iterated convexification. It is however very time consuming to compute.

7.3 Similarity as a Means of Prediction

We discussed the use of confidence measures as a means of predicting the classification of a new example in Section 5.1, and saw that for target functions which could be represented by DNFs with a small number of monomials, each with only a small number of literals in it, we could use similarity to the positive (and negative) examples in the training sample to predict the classification of a new example. We now present experimental results that show the results of using the similarity of a new example to the existing examples in this way.

The training sample A is split into the set of positive examples A^+ and the set of negative examples A^-. For each element x in the corresponding training set, we computed both $s(x,A^+)$ and $s(x,A^-)$, and the difference between them $s(x,A^+) - s(x,A^-)$. Call this the *similarity classification*. We then

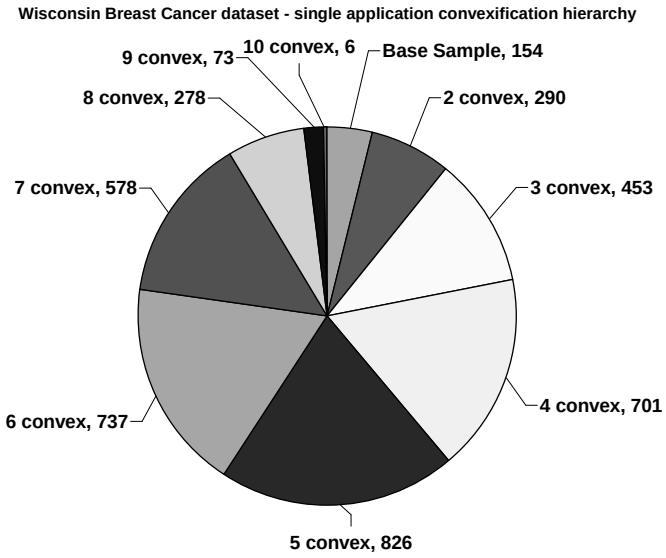


Figure 10: Wisconsin Breast Cancer Data: Single Application k-Convexification Hierarchy

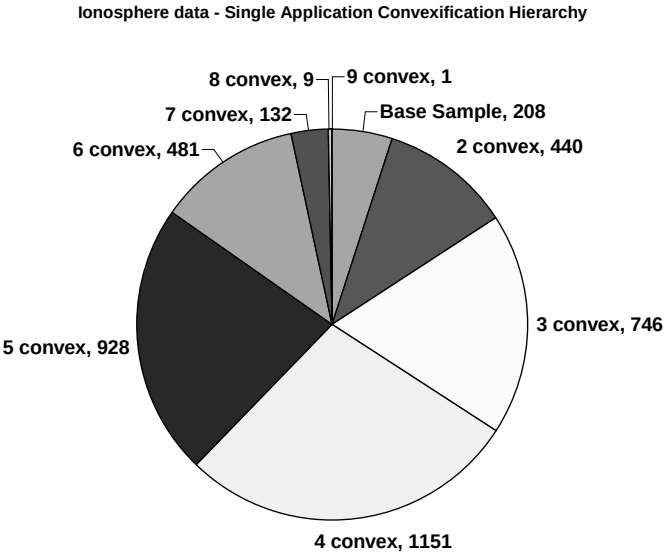


Figure 11: Ionosphere Data: Single Application k-Convexification Hierarchy

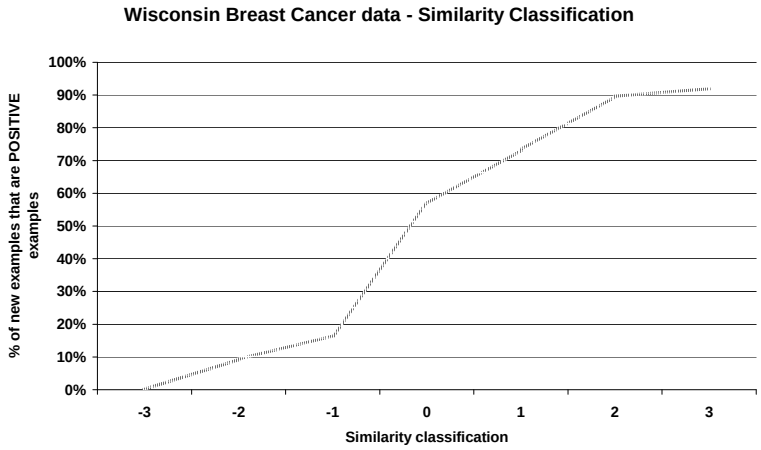


Figure 12: Wisconsin Breast Cancer Data: Similarity Classification

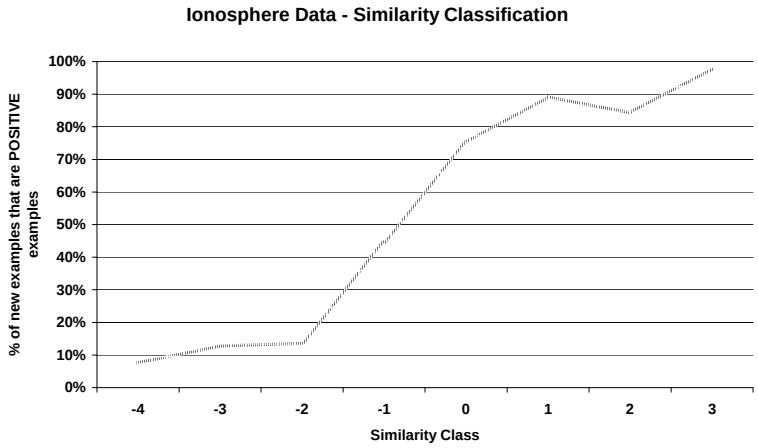


Figure 13: Ionosphere Data: Similarity Classification

looked at the ratio of true examples with each similarity classification to the number of all examples with that classification. Figure 12 on page 40 and Figure 13 on page 40 show that examples with a high similarity classification (i.e. that are much more similar to A^+ than to A^-) are much more likely to be positive examples.

7.4 Convexity as a Means of Prediction

In an analogous way to the use of similarity for prediction in the previous section, we now consider whether examples that are more closely *flanked* by positive examples than negative examples are themselves more likely to be positive examples.

We split each training sample A into the set of positive examples A^+ and the set of negative ex-

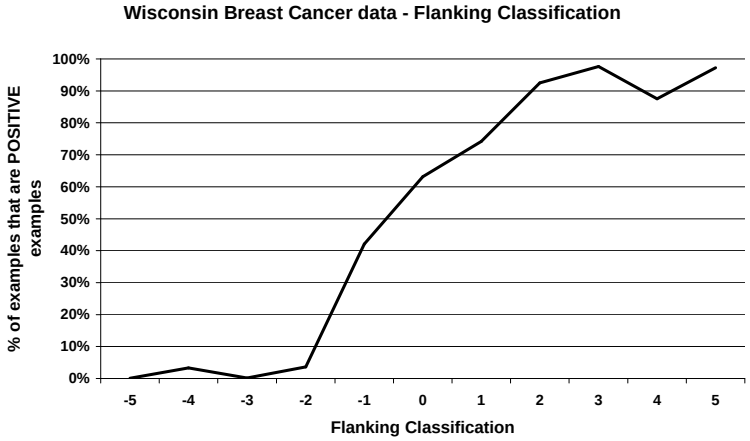


Figure 14: Wisconsin Breast Cancer Data: Flanking Classification

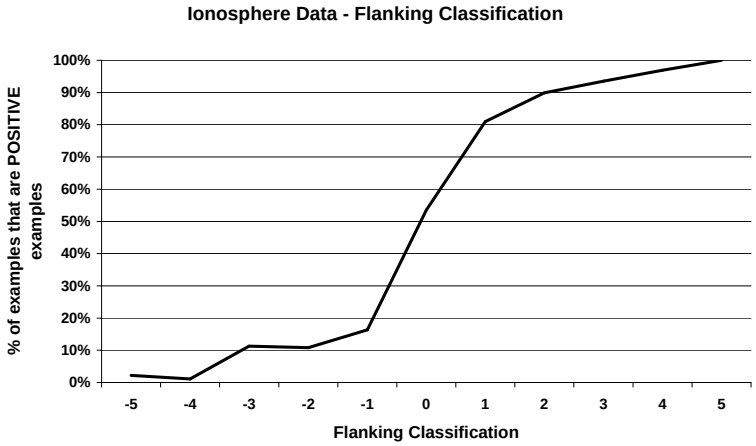


Figure 15: Ionosphere Data: Flanking Classification

amples A^-. For each element x in the corresponding training set, we computed the flanking distances $\mathbb{F}(x,A^+),\mathbb{F}(x,A^-)$. We then calculated the difference between the two flanking distances (call this the *flanking classification*) as $\mathbb{FC}(x) = \mathbb{F}(x,A^-) - \mathbb{F}(x,A^+)$. Recall that low flanking distance implies that an example is "close" to the corresponding sample, and so this flanking classification will be large and positive if x is closely flanked by the positive examples and not closely flanked by the negative examples. It will be large and negative if x is closely flanked by the negative examples and not closely flanked by the positive examples. We then looked at the ratio of true examples with each flanking classification to the number of all examples with that classification. Figure 14 on page 41 and Figure 15 on page 41 clearly show that in general, as $\mathbb{FC}(x)$ increases, the example is much more likely to be a positive example, as expected.

8 Conclusions

The four methods of estimating confidence outlined in this paper are all quite different, and their degree of usefulness varies by dataset and in particular by the type of the target function t that we are trying to learn. There is no single measure of how much an element x is "like" A, and indeed the fact that x can be "like" A in one way and unlike A in another suggests that there are likely to be other useful confidence measures, measuring different aspects of the relationship between x and A, in addition to the measures discussed here.

If we can assume that the target function is of a particular type, we can choose one of the measures as particularly appropriate. For example, where t can be represented as a DNF with a small number of literals in each term ($\leq k$, say), and a small number of terms ($\leq d$, say), similarity is particularly useful, for the reasons outlined in Section 5.1. On the other hand, when t is a k-convex function, as discussed by Ekin et al. (1998), flanking distance is likely to prove more useful. In fact in both these cases, given a simple enough function t and enough data, we can sometimes be 100% confident in our classification.

Unfortunately, none of the measures outlined in this paper allow us to say in all cases that the fact that an example x is more "like" our training sample A, as defined by the measure, than another example y, means that the classification of x by our hypothesis h is more likely to be accurate than the classification of y by h; indeed Veal's work shows that in some cases similarity can be misleading. This is not a particularly surprising conclusion in view of our reluctance to make any probabilistic assumptions about the distribution of the examples; we saw in Section 2.2 that with this type of assumption, one can make quite strong statements about the appropriate degree of confidence in a hypothesis. In the absence of such assumptions, we might better call our confidence measures 'confidence heuristics' - they *suggest* that classification is more likely to be accurate on some examples than others, but do not guarantee it. On the positive side, while the theoretical support for the confidence measures is not as strong as might be desired, the experimental results presented here suggest that for every confidence measure discussed, we could almost always attach a higher degree of confidence to the classification of a example x with a very high value for the measure than to the classification of another example with a very low value for the measure. Further, in almost all cases, if x has a higher value for the measure than y does, the expected accuracy of the classification of x will be at least as high as that of y.

We note that given a training sample $A \subseteq \{0,1\}^n$, split into positive examples A^+ and negative examples A^-, how much a new example $x \in \{0,1\}^n$ is "like" A^+ and A^- is often of more interest than how much it is "like" A as a whole.

A number of interesting areas for further work are suggested below.

8.1 Further Work

8.1.1 Errors in Input Data

Apart from the work on outliers, this paper assumes that we are dealing with data sets that are error-free. When considering real data sets this is an unrealistic assumption and an obvious question is how learning in general, and confidence measures in particular, are affected by errors in the input data. It is possible to PAC learn from noisy data (i.e. data with errors), as shown by for example Kearns (1998). But Algorithm 6.2 on page 24 will not work correctly on noisy data without modification. Recall that the hypothesis produced by this algorithm is the disjunction of a number of monomials $M = \{m_i\}$, each of which has been seen in some positive examples - and no negative examples - in a training sample A for a target function t. If each example represents a patient, each monomial in the hypothesis can be thought of as representing a group of symptoms (a "syndrome") that indicates the presence of the disease in question. But what if the training sample contains an example which satisfies some monomial $m_j \in M$, but is wrongly classified as a negative example? The learning algorithm will remove m_j from h, even though it may be true on a very large number of positive examples. We propose the following modification, which it would be interesting to study further, based on work by Boros et al. (1996).

Consider any monomial m. We can construct two subsets $X^+, X^- \subseteq A$, where $X^+ = \{x \in A : m(x) = 1, t(x) = 1\}$ (the positive examples in A that satisfy m) and $X^- = \{x \in A : m(x) = 1, t(x) = 0\}$ (the negative examples in A that satisfy m). Now there are four possible cases:

1. $X^+ = X^- = \emptyset$. There are no elements of A that satisfy m. In this case, m should not be in the hypothesis (indeed it is clearly in one of the similarity functions d_i defined in Algorithm 6.2 on page 24).

2. $X^+ \neq \emptyset, X^- = \emptyset$. m is satisfied only by positive examples in A. It should appear in the hypothesis, as indeed it would with the unmodified algorithm.

3. $X^+ = \emptyset, X^- \neq \emptyset$. m is satisfied only by negative examples in A. It should not appear in the output hypothesis or the d_i functions (again this is the same as the unmodified algorithm).

4. $X^+ \neq \emptyset, X^- \neq \emptyset$. m is satisfied by both positive and negative examples in A. Boros et al. suggest that m is assumed to represent a positional substring that should lead to an example being classified as positive if $\frac{|X^-|}{|X^+|} \leq \tau$, for some τ. They note that τ is a "problem dependent parameter" and suggest, based on experimental evidence, that τ should be between 0 and 0.2.

This deals with the output hypothesis. But how should one calculate similarity with noisy data? Most work on noise in data distinguishes two types of noise: errors in the classification of an example (called "classification noise") and errors in the inputs or attributes for an item, called "attribute noise".

If an example has the first type of noise, it will be shown in the training sample as being of type Y whereas it is actually of type $Z \neq Y$. If it has the second type of noise, some of the attributes of the example will be wrong (for example, if an example represents a patient, the patient might be shown in the dataset as male when she is actually female, or wrongly shown as not having a certain symptom when she does in fact have it). While we have in this paper often considered similarity to the positive and negative examples in a dataset separately, the most basic use of similarity does not depend on the classification of elements of the training sample, which means that classification noise would have no effect. However, since the presence of attribute noise will change the positional substrings present in each example, and therefore potentially those present in the dataset as a whole, this type of noise is clearly more of a problem for calculating similarity. At present, a dataset is classified as containing any given positional substring s, for the purposes of calculating similarity, if there is at least one element of the dataset which contains s. One obvious method, by analogy with the method presented above for the output hypothesis, would be to require at least a certain percentage $\alpha > 0$ of the dataset to contain s before we were willing to classify it as present. The obvious question is "what is a reasonable value of α?". We leave this as a question for further study, except to note that any rigorous derivation is likely to involve some probabilistic assumptions about the relative frequency of attribute values.

8.1.2 Confidence Measures for Other Types of Boolean Function

There are many different types of Boolean function that have been studied in the literature, including l-term k-DNFs, k-convex functions, threshold functions and Horn clauses (Horn (1951)). Similarity and convexity seem particularly suited to estimating confidence when the target function is an l-term k-DNF, or a k-convex function respectively. Are there confidence measures that would be well suited for the other classes?

8.1.3 Relationship between k-convex Functions and l-term j-DNFs

We considered earlier two classes of Boolean functions, k-convex functions $\Phi(n,k)$ and l-term j-DNFs (written $D^l_{n,j}$). What is the degree of overlap between the two classes of function? Let us be more specific. Suppose we have a function $f \in \Phi(n,k)$. Can we find bounds $j_0, l_0 \in \mathbb{Z}^+$ such that $f \in D^l_{n,j} \forall j \geq j_0, l \geq l_0$? Further work is needed here, although Ekin et al. (1998) did show that where $k > \frac{n}{2} - 1$, we can write f as a DNF with at most $\frac{2(k+1)}{2(k+1)-n}$ terms, which gives us a bound for l when k is large.

8.1.4 Prevalence Count and Structure

High prevalence count does seem to correlate with high accuracy, but we have not explored this in detail. The experimental results suggest that there is a threshold prevalence count (call this γ) above which classification is accurate, and below which it is not. Is this simply an artifact of the particular datasets used? Is there some way to calculate or at least bound γ for each dataset, ideally based only on the size of the dataset and the number of inputs? Hammer & Bonates (2005) do discuss what they call "strong positive patterns", which correspond to high prevalence monomials, but do not attempt to quantify "strong". It may be that the assumption that examples are presented based on a fixed but unknown probability distribution is needed to be able to say anything meaningful here.

8.1.5 Outliers

In Section 5.2 we looked at using similarity to classify certain elements of the dataset as outliers. Experimental results were disappointing; does this mean that similarity is not useful for identifying outliers? Further work is needed on the theoretical justification for using similarity in this way. The use of convexity to identify possible outliers would also be worth investigating.

8.1.6 Use of Confidence Measures with other Learning Algorithms

This paper has considered a variety of confidence measures, but only used one learning algorithm (Algorithm 6.2 on page 24). Are the confidence measures presented here useful in conjunction with other algorithms? Anthony, Hammer, Subasi and Subasi (in preparation) have investigated using similarity with a range of learning algorithms available as part of the WEKA package (Witten & Frank 2005), and the results appear to be similar to the results obtained with the learning algorithm presented in this paper.

8.1.7 k-convex Hulls

In Section 4.3.1 we conjectured that a sufficient condition for the convex hull of a set A to be the whole of $\{0,1\}^n$ is that $\forall i = 1,\ldots,n$ we can find $x^+, x^- \in A$ such that $x_i^+ = 1$ and $x_i^- = 0$, with $d(x^+, x^-) \leq k$, and that $n \leq \beta(k)$ for some unknown function β. We know it is true in the trivial case $k = n$. Is this true for $k < n$? If so, what is the function β? Are there simpler conditions we could use?

8.2 Acknowledgements

Thanks to Martin Anthony and Ben Veal for many helpful comments and suggestions, including pointing out that the complement of an l-term DNF can be written as a DNF with at most l literals in each term, i.e. a member of $D_{n,l}$.

References

Anthony, M. H. G., and Biggs, N. L. (1992), *Computational Learning Theory*, Cambridge: Cambridge University Press.

Anthony, M. H. G., Brightwell, G., and Shawe-Taylor, J. (1995). On Specifying Boolean Functions by Labelled Examples. Discrete Applied Mathematics **61**, 1-25.

Anthony, M. H. G., and Hammer, P. L. (2004). A Boolean Measure of Similarity. Technical Report 27-2004, Rutgers Center for Operational Research.

Anthony, M. H. G., and Hammer, P. L. (2005). Defining Domains of Confidence in Machine Learning. Personal Communication.

Barber, C. B., Dobkin, D. P., and Huhdanpaa, H. (1996). The Quickhull Algorithm for Convex Hulls. ACM Trans. Math. Softw. **22**, 469-83.

Boros, E., et al. (1996). An Implementation of Logical Analysis of Data. Technical Report 22-96, Rutgers Center for Operational Research.

Cormen, T. T., Leiserson, C. E., Rivest, R. L. (1990). *Introduction to algorithms*,MIT Press, Cambridge, MA, USA.

Ekin, O., Hammer, P. L., and Kogan, A. (1998) Convexity and Logical Analysis of Data. Technical Report 5-98, Rutgers Center for Operational Research.

Hammer, P. L., and Bonates, T. (2005). Logical Analysis of Data: From Combinatorial Optimization to Medical Examples. Technical Report 10-2005, Rutgers Center for Operational Research.

Hettich, S., Blake, C. L., and Merz, C. J. (1998), *UCI Repository of Machine Learning Databases*. University of California at Irvine. Available from <http://www.ics.uci.edu/∼mlearn/MLRepository.html> [Accessed 22 August 2005].

Horn, A. (1951) On Sentences which are True of Direct Unions of Algebras. Journal of Symbolic Logic, **16**, 14-21.

Kearns, M. (1998) Efficient Noise-Tolerant Learning from Statistical Queries. J. ACM **45**, 983-1006.

Kivinen, J. (1995). Learning Reliably and with One-Sided Error. Mathematical Systems Theory **28**, 141-172.

Kruger, M. Wille, C., Holm, C. Spuida, B. (2000) *SharpDevelop: Open Source Development Environment for C#* [Internet] Available from <www.icsharpcode.net> [Accessed 22 August 2005].

Rivest, R. L., and Sloan, R. (1988). Learning Complicated Concepts Reliably and Usefully. *COLT '88: Proceedings of the first annual workshop on Computational learning theory*, San Francisco, California: Morgan Kaufmann, pp. 69-79.

Valiant, L. G. (1984). A Theory of the Learnable. Commun. ACM **27**, 1134-1142.

Veal, B. (2005a). Properties of a Binary Similarity Measure. Technical Report LSE-CDAM-2005-06, Centre for Discrete and Applicable Mathematics (CDAM), London School of Economics.

Veal, B. (2005b) *Binary Predict* software (includes source code for algorithms presented in this paper) [Internet] Available from http://www.maths.lse.ac.uk/Personal/ben/

Vovk, V. (2002) Asymptotic optimality of Transductive Confidence Machine. Proceedings of the 13th International Conference on Algorithmic Learning Theory, 2002, Lecture Notes in Artificial Intelligence **2533**, 336-350.

Witten, I. H. and Frank, E. (2005). *Data Mining: Practical machine learning tools and techniques, 2nd Edition*. San Francisco, Morgan Kaufmann. WEKA software available from <http://www.cs.waikato.ac.nz/ml/weka/> [Accessed 22 August 2005].

A Counterexample for k-convex Hulls

We present a set of points in $\{0,1\}^{20}$ such that for $i = 1,\ldots,20$ we can find x_i^+, x_i^- with bit i of $x_i^+ = 1$ and bit i of $x_i^- = 0$, and $d(x_i^+, x_i^-) \leq 2$, but where the 2-convex hull of the points is not the whole of $\{0,1\}^{20}$. For each $j = 1,\ldots,20$ take points p_{2j-1} and p_{2j}. They are distance 1 apart. Form the convex hull of each pair and then the disjunction of all those convex hulls. As can be verified, this gives the DNF below. Since each pair of terms in the DNF conflicts in at least 3 literals, this DNF represents a 2-convex function and so will not be changed by further applications of 2-convexification.

$$
\begin{aligned}
& u_1u_2u_3u_4u_5u_6u_7u_8u_9u_{10}u_{11}u_{12}u_{13}u_{14}u_{15}u_{16}u_{17}u_{18}u_{19} \\
\vee\ & \overline{u}_1\overline{u}_2\overline{u}_3u_4u_5u_6u_7u_8u_9u_{10}u_{11}u_{12}u_{13}u_{14}u_{15}u_{16}u_{17}u_{18}u_{20} \\
\vee\ & u_1u_2u_3\overline{u}_4\overline{u}_5\overline{u}_6u_7u_8u_9u_{10}u_{11}u_{12}u_{13}u_{14}u_{15}u_{16}u_{17}u_{19}u_{20} \\
\vee\ & u_1u_2u_3u_4u_5u_6\overline{u}_7\overline{u}_8\overline{u}_9u_{10}u_{11}u_{12}u_{13}u_{14}u_{15}u_{16}u_{18}u_{19}u_{20} \\
\vee\ & u_1u_2u_3u_4u_5u_6u_7u_8u_9\overline{u}_{10}\overline{u}_{11}\overline{u}_{12}u_{13}u_{14}u_{15}u_{17}u_{18}u_{19}u_{20} \\
\vee\ & u_1u_2u_3u_4u_5u_6u_7u_8u_9u_{10}u_{11}u_{12}\overline{u}_{13}\overline{u}_{14}\overline{u}_{16}u_{17}u_{18}u_{19}u_{20} \\
\vee\ & u_1u_2u_3u_4u_5\overline{u}_6u_7u_8u_9u_{10}u_{11}u_{12}u_{13}u_{15}u_{16}\overline{u}_{17}\overline{u}_{18}\overline{u}_{19}u_{20} \\
\vee\ & u_1u_2u_3u_4u_5u_6u_7u_8u_9u_{10}u_{11}\overline{u}_{12}u_{13}u_{15}\overline{u}_{16}u_{17}u_{18}\overline{u}_{19}\overline{u}_{20} \\
\vee\ & u_1u_2u_3u_4u_5u_6u_7u_8u_9u_{10}u_{11}u_{13}u_{14}\overline{u}_{15}\overline{u}_{16}\overline{u}_{17}\overline{u}_{18}u_{19}u_{20} \\
\vee\ & \overline{u_1}u_2u_3\overline{u}_4u_5u_6u_7u_8u_9u_{10}u_{12}\overline{u}_{13}u_{14}\overline{u}_{15}\overline{u}_{16}u_{17}u_{18}u_{19}u_{20} \\
\vee\ & u_1\overline{u_2}u_3u_4u_5u_6u_7u_8u_9u_{11}u_{12}u_{13}\overline{u}_{14}u_{15}\overline{u}_{16}\overline{u}_{17}u_{18}u_{19}u_{20} \\
\vee\ & u_1u_2\overline{u}_3\overline{u}_4u_5u_6\overline{u}_7u_8u_{10}u_{11}u_{12}u_{13}u_{14}u_{15}u_{16}u_{17}u_{18}u_{19}u_{20} \\
\vee\ & u_1u_2u_3\overline{u}_4u_5u_6u_7\overline{u}_9\overline{u}_{10}u_{11}u_{12}u_{13}u_{14}u_{15}u_{16}u_{17}u_{18}u_{19}u_{20} \\
\vee\ & u_1u_2u_3u_4\overline{u}_5u_6u_8u_9u_{10}\overline{u}_{11}u_{12}\overline{u}_{13}u_{14}u_{15}u_{16}u_{17}u_{18}u_{19}u_{20} \\
\vee\ & u_1u_2u_3u_4u_5\overline{u}_7u_8u_9\overline{u}_{10}u_{11}u_{12}u_{13}\overline{u}_{14}u_{15}u_{16}u_{17}u_{18}u_{19}u_{20} \\
\vee\ & u_1u_2u_3u_4\overline{u}_6\overline{u}_7u_8u_9u_{10}u_{11}\overline{u}_{12}u_{13}u_{14}u_{15}u_{16}u_{17}u_{18}u_{19}u_{20} \\
\vee\ & u_1u_2u_3u_5u_6u_7\overline{u}_8u_9u_{10}u_{11}u_{12}\overline{u}_{13}u_{14}u_{15}u_{16}\overline{u}_{17}u_{18}u_{19}u_{20} \\
\vee\ & \overline{u}_1u_2u_4u_5u_6u_7u_8\overline{u}_9\overline{u}_{10}u_{11}u_{12}u_{13}\overline{u}_{14}u_{15}u_{16}u_{17}u_{18}u_{19}u_{20} \\
\vee\ & u_1u_3u_4u_5u_6u_7u_8u_9u_{10}\overline{u}_{11}u_{12}u_{13}u_{14}u_{15}u_{16}u_{17}\overline{u}_{18}\overline{u}_{19}u_{20} \\
\vee\ & u_2u_3\overline{u}_4u_5\overline{u}_6u_7u_8u_9u_{10}u_{11}\overline{u}_{12}u_{13}u_{14}\overline{u}_{15}u_{16}u_{17}u_{18}u_{19}u_{20}
\end{aligned}
$$

The set of points $\{p_i\}$ is:

p_1 (11111111111111111110) p_2 (11111111111111111111)
p_3 (00011111111111111101) p_4 (00011111111111111111)
p_5 (11100011111111111011) p_6 (11100011111111111111)
p_7 (11111100011111110111) p_8 (11111100011111110111)
p_9 (11111111100011101111) p_{10} (11111111100011111111)
p_{11} (11111111111100001111) p_{12} (11111111111100101111)
p_{13} (11111011111110110001) p_{14} (11111011111111110001)
p_{15} (11111111111001101100) p_{16} (11111111111011101100)
p_{17} (11111111111011000011) p_{18} (11111111111111000011)
p_{19} (01101111110101001111) p_{20} (01101111111101001111)
p_{21} (10111111101110100111) p_{22} (10111111111110100111)
p_{23} (11001101011111111111) p_{24} (11001101111111111111)
p_{25} (11101110001111111111) p_{26} (11101111001111111111)
p_{27} (11110101110101111111) p_{28} (11110111110101111111)
p_{29} (11111001101110111111) p_{30} (11111101101110111111)
p_{31} (11110001111011111111) p_{32} (11111001111011111111)
p_{33} (11101110111101110111) p_{34} (11111110111101110111)
p_{35} (01011111001110111111) p_{36} (01111111001110111111)
p_{37} (10111111110111111001) p_{38} (11111111110111111001)
p_{39} (01101011111011011111) p_{40} (11101011111011011111)

B Explanation of Key Monomial in Hypotheses on Voting Records

In Section 7.1.4 we found that a single monomial of length 2 ($u_4\overline{u}_{11}$) was satisfied by 83% of positive examples and no negative examples in the Voting Records dataset. We investigate here what this monomial actually *means* when we relate it back to what the dataset represents.

The dataset shows how each member of the U.S. House of Representatives voted in the 16 key votes identified by the 1984 Congressional Quarterly Almanac. Each member of the House of Representatives is either a member of the Republican party or of the Democratic Party. Our aim was to find a hypothesis which accurately predicted the party of a member based on their voting record; arbitrarily, we classify Republicans as type 1 and Democrats as type 0.

The hypothesis h generated by Algorithm 6.2 on page 24 for the Voting Records dataset was the disjunction of several monomials, including $u_4\overline{u}_{11}$ which we will call m. Monomial m would be satisfied by example x if the person represented by x had voted "yes" in vote 4 and "no" in vote 11. Vote number 4 was to extend a pay freeze for physicians (medical doctors). Republicans almost all voted "yes", with Democrats strongly against. Vote number 11 was on funding for research into synthetic fuel (substitutes for crude oil). The motion was that funding should be reduced by a certain amount; a vote against this motion was generally taken as a desire to reduce funding by even more. Most (although by no means all) Republicans voted "no", while Democrats mostly voted "yes". What m implies (and this can easily be verified by looking at the dataset) is that anyone who voted for a physician pay freeze and against a relatively small reduction in funding for research into synthetic fuels *must* be a Republican. By checking the dataset, we can see that 83% of Republicans voted in this way.

We can also ask what the accuracy of m is on the dataset as a whole. If we take our hypothesis h_m to be "IF X voted FOR motion 4 AND AGAINST motion 11 THEN X is a Republican ELSE X is a Democrat", we find that it has an accuracy of 92.2%. The accuracy is higher than the percentage of Republicans whose voting patterns match the rule because the dataset contains more Democrats than Republicans, and if X is a Democrat the rule will always classify them correctly. Recall that the unmodified h had an accuracy in the region of 70%, and that to achieve accuracy of above 90% we had to either disregard low prevalence monomials or to remove some of the unnecessary inputs. This shows the importance of the prevalence count as a measure of confidence.