II Boolean Formulae and Representations.

1. Representing Monomial Concepts
2. The Standard Learning Algorithm for Monomials (FIND-S)
3. Standard Notation for Boolean Functions
4. Boolean Hypothesis Spaces
5. Learning Disjunctions of Monomials
6. Summary of Lecture

Introduction

This lecture is based on chapter 2 of Anthony and Biggs. We introduce notation, hypothesis classes, and algorithms which we will use later in the course. Specifically:

- the class of monomial concepts and notation for it.
- standard notation (DNF and CNF) for Boolean functions.
- some spaces of hypotheses: $B_n, M_n, M_{n,k}, D_{n,k}$.
- learning algorithms for monomials and disjunctions of small monomials.

1 Representing Monomial Concepts

In lecture 1 we saw that we can represent concepts (functions) in different ways. For example, one way of representing the logical AND function is using definition by cases:

$$c(y) = \begin{cases} 1 & \text{if } y_1 \text{ and } y_2 \text{ are 1;} \\ 0 & \text{otherwise.} \end{cases}$$

and another is using a truth table:

y_1	y_2	$c(y)$
0	0	0
0	1	0
1	0	0
1	1	1

Variables and Literals

We say that a function which accepts a string of n bits is a function of n *variables*.

u_i refers to the value of variable i and $\bar{u}_i$ refers to the negation of this variable.

For example, for the string 0010, $u_3 = 1$ and $\bar{u}_3 = 0$.

We call the functions $u_1\bar{u}_1 \ldots u_n\bar{u}_n$ *literals*.

Monomial Formulae

We can represent a subset of functions, called *monomial functions*, in a third way, using strings of *literals* $u_1, \bar{u}_1, \ldots$ called *monomial formulae*.

Here we represent the logical AND using both a truth table and a monomial formula:

y_1	y_2	$c(y)$
0	0	0
0	1	0
1	0	0
1	1	1

$= \langle u_1 u_2 \rangle$

In the monomial formula, u_1 indicates that for the output to be 1, y_1 must be 1. Similarly, u_2 indicates that y_2 must be 1. The literals in the string are implicitly ANDed together, so both u_1 AND u_2 must be 1 for the output to be 1. The brackets $\langle\rangle$ indicate the monomial formula is a function (a concept).

Monomial Formulae (Continued)

To indicate that a bit must be 0, we add a bar to the corresponding literal, i.e., $\bar{u}_i$. For example,

y_1	y_2	$c(y)$
0	0	0
0	1	1
1	0	0
1	1	0

$= \langle \bar{u}_1 u_2 \rangle$

Monomial Formulae (Continued)

Finally, to indicate that a bit can take either value we omit the corresponding literal. For example,

y_1	y_2	$c(y)$
0	0	0
0	1	0
1	0	1
1	1	1

$= \langle u_1 \rangle$

Note: Many Boolean functions (e.g., the parity function) are not monomial, and cannot be represented by a monomial formula.

2 The Standard Learning Algorithm for Monomials (FIND-S)

Suppose objects are described in terms of the following properties:

1. Four legs
2. Tail
3. Flat seat
4. Coloured brown
5. Alive

so that the example $(0, 1, 0, 1, 1)$ denotes something which has a tail, is brown, and is alive.

Generalising Over-specific Hypotheses

Now suppose that we are trying to learn the concept ‘chair’ from examples, and that the current hypothesis is $h = \langle u_1 \bar{u}_2 u_3 u_4 \bar{u}_5 \rangle$, i.e., that chairs have 4 legs, no tails, flat seats, are brown and are not alive.

If we next see a positive example of a chair which is not brown (e.g., $(10100, 1)$), we can revise our hypothesis so that brownness is irrelevant. I.e., we revise h such that $h = \langle u_1 \bar{u}_2 u_3 \bar{u}_5 \rangle$.

This demonstrates the principle behind the standard learning algorithm for monomials: that we can generalise an overly specific hypothesis when we find counterexamples.

The Standard Algorithm for Monomials

The standard algorithm from page 11 of Anthony and Biggs is:

1. Set $U = \{u_1, \bar{u}_1, \ldots, u_n, \bar{u}_n\}$
2. For each example i
3. If i is positive then
4. For each bit j in i
5. If $j = 1$ then delete $\bar{u}_j$ if present in U
6. Else delete u_j if present in U
7. Return the hypothesis whose monomial formula contains the literals in U

Notes:

- This algorithm is a memoryless on-line algorithm.
- Mitchell (section 2.4) calls this the FIND-S algorithm.

An Example of the Standard Algorithm

Suppose there are $n = 3$ literals representing the following properties:

1. Tail
2. Whiskers
3. Black

and that the target concept “cat” is $\langle u_1 u_2 \rangle$, i.e., that cats have tails and whiskers. Suppose the training sample is:

$$((101, 0), (111, 1), (110, 1)).$$

According to line 1, the initial hypothesis is $\langle u_1 \bar{u}_1 u_2 \bar{u}_2 u_3 \bar{u}_3 \rangle$.

The Example, in Detail

Example $(101, 0)$ is ignored (line 3) since it is negative; it is not an example of a cat, and so tells us nothing about them. (Perhaps it is an example of a dog.) The current hypothesis is still $\langle u_1 \bar{u}_1 u_2 \bar{u}_2 u_3 \bar{u}_3 \rangle$.

Example $(111, 1)$ is a cat, and has a tail, whiskers, and is black. So we delete the literals which state that cats do not have tails ($\bar{u}_1$), do not have whiskers ($\bar{u}_2$), and are not black ($\bar{u}_3$). The current hypothesis is $\langle u_1 u_2 u_3 \rangle$, i.e., that cats have tails, whiskers, and are black (remember, we’ve only seen one cat, and it was black, so it is not unreasonable to assume all cats are black).

Finally, example $(110, 1)$ is a cat which is not black, which tells us cats are not necessarily black. We delete u_3 so that the current hypothesis is now identical to the target hypothesis $\langle u_1 u_2 \rangle$.

How Many Examples does the Standard Algorithm Need?

The standard algorithm specifies the identically-zero function†, which we can consider a fully-specific hypothesis, as the initial hypothesis. It then generalises this hypothesis by removing literals when it finds counterexamples to them in examples of the target concept.

Clearly, if all positive examples of the target concept are in the training sample, the final hypothesis will be correct. In terms of the example, we will have seen all possible cats, as indeed we did.

However, it is not always necessary for all positive examples to be seen, as the example on page 11 of Anthony and Biggs shows.

† The function which always returns 0.

3 Standard Notation for Boolean Functions

Monomials

Monomial formulae are conjunctions of literals; strings of literals connected by logical ANDs:

$$u_1 \wedge u_2 \wedge \ldots \wedge u_n$$

By convention we omit the $\wedge$ when its meaning is clear. I.e., $u_1 \wedge u_2 \wedge \ldots \wedge u_n \;=\; u_1 u_2 \ldots u_n$

Clauses

Clauses are disjunctions of literals; strings of literals connected by logical ORs:

$$u_1 \vee u_2 \vee \ldots \vee u_n$$

Normal Forms

There are many ways to combine literals, but we’ll use two normal forms:

Disjunctive Normal Form (DNF)

An expression in DNF is a disjunction of monomials:

$$\mu_1 \vee \mu_2 \vee \ldots \vee \mu_r$$

Conjunctive Normal Form (CNF)

An expression in CNF is a conjunction of clauses:

$$\gamma_1 \wedge \gamma_2 \wedge \ldots \wedge \gamma_s$$

Normal Forms (Continued)

Any Boolean function can be written in both DNF and CNF (see page 17 of Anthony and Biggs).

Remember:

- DNF is a disjunction of conjunctions.
- CNF is a conjunction of disjunctions.

4 Boolean Hypothesis Spaces

The space of all Boolean functions can be divided a number of ways. E.g.:

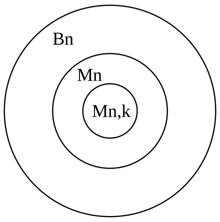
- B_n The set of all Boolean functions of n variables.
- M_n The set of functions represented by monomials defined on n variables.
- $M_{n,k}$ The set of functions of n variables represented by monomials with at most k literals.
- $D_{n,k}$ The set of functions represented by disjunctions of monomials in $M_{n,k}$.

Examples

$M_2 = \{u_1,\ u_2,\ u_1u_2,\ \bar{u}_1,\ \bar{u}_2,\ \bar{u}_1\bar{u}_2,\ \bar{u}_1u_2,\ u_1\bar{u}_2\}$

$M_{2,1} = \{u_1,\ u_2,\ \bar{u}_1,\ \bar{u}_2\}$

$D_{2,1} = \{u_1,\ u_2,\ \bar{u}_1,\ \bar{u}_2,\ u_1 \vee u_2,\ \bar{u}_1 \vee \bar{u}_2,\ \bar{u}_1 \vee u_2,\ u_1 \vee \bar{u}_2\}$



Note:

- $B_n = \{0,1\}^n \rightarrow \{0,1\}$ (all Boolean functions of n variables)
- $M_n \subset B_n$
- $M_{n,k} \subseteq M_n$

5 Learning Disjunctions of Small Monomials

An algorithm similar to the standard algorithm for monomials can be used to learn disjunctions of small monomials.

1. Set h = disjunction of all monomials of length at most k literals
2. For each example i
3. If i is negative and $h(i)$ is positive
4. Then delete monomials μ from h for which $\mu(i)$ is positive

An Example

Suppose we are working in $D_{3,2}$, the target concept is $\langle \bar{u}_1 \vee (u_2\bar{u}_3)\rangle$, and the three negative examples are 101, 111 and 100.

The initial hypothesis is $D_{3,2}$, the disjunction of all monomial formulae in $M_{3,2}$:

$$h = \langle u_1 \vee u_2 \vee u_3 \vee \bar{u}_1 \vee \bar{u}_2 \vee \bar{u}_3 \vee u_1u_2 \vee u_1u_3 \vee u_2u_3 \vee \bar{u}_1u_2 \vee \bar{u}_1u_3$$
$$\vee\ \bar{u}_2u_3 \vee u_1\bar{u}_2 \vee u_1\bar{u}_3 \vee u_2\bar{u}_3 \vee \bar{u}_1\bar{u}_2 \vee \bar{u}_1\bar{u}_3 \vee \bar{u}_2\bar{u}_3\rangle$$

example	monomials deleted
101	$u_1, u_3, \bar{u}_2, u_1u_3, \bar{u}_2u_3, u_1\bar{u}_2$
111	u_2, u_1u_2, u_2u_3
100	$\bar{u}_3, u_1\bar{u}_3, \bar{u}_2\bar{u}_3$

The final, correct, hypothesis is:

$$h = \langle \bar{u}_1 \vee \bar{u}_1u_2 \vee \bar{u}_1u_3 \vee u_2\bar{u}_3 \vee \bar{u}_1\bar{u}_2 \vee \bar{u}_1\bar{u}_3\rangle$$

Although this is much longer than $\langle \bar{u}_1 \vee (u_2\bar{u}_3)\rangle$, the two are equivalent.

6 Summary of Lecture

- Literals are functions which return the value u_i of a variable i or its negation $\bar{u}_i$.

Monomials and Clauses

- We can represent Monomial functions using monomial formulae, which are conjunctions (ANDs) of literals $u_1u_2\ldots u_n$.
- Clauses are disjunctions (ORs) of literals $u_1u_2\ldots u_n$.
- The standard algorithm for monomials starts with all literals and deletes those which are falsified by positive examples.

Summary (Continued)

Arbitrary Boolean Functions

- An expression in Disjunctive Normal Form (DNF) is a disjunction of monomials $\mu_1 \vee \mu_2 \vee \ldots \vee \mu_r$.
- An expression in Conjunctive Normal Form (CNF) is a conjunction of clauses $\gamma_1 \wedge \gamma_2 \wedge \ldots \wedge \gamma_s$.
- Some Boolean hypothesis spaces: $B_n, M_n, M_{n,k}, D_{n,k}$.
- The algorithm for learning disjunctions of small monomials starts with the disjunction of all monomials and deletes those which are falsified by negative examples.