On an isomorphism of n-bit functions and their number

Mark R. Diamond
Daniel D. Reidpath

Abstract

In this paper, we describe a useful isomorphism of n-bit functions and present a count of the number of non-isomorphic forms for small n.

1 A 1-bit illustration

Imagine creating a system for deciding which of the many thousands of people who visit Australia each year, you should search for drugs. You start with something simple, classifying all individuals according to whether or not they visited the Golden Triangle on their way to Australia. In particular, for each visitor you assign to the variable x_1 the value 1 if that visitor visited the Golden Triangle, and the value 0 if they did not.

Subsequently you thoroughly search every visitor and determine their status as a smuggler, recording your determination as variable y to which you assign the value 1 if the visitor is a smuggler, and 0 if they are 'clean'. We assume that both your classification and your coding are error-free.

You submit your collection of visitor profiles to your learning machine and it discovers the relationship shown in Table 1(a). Since the direction of

x_1	y
0	0
1	1

(a)

x_1	y
0	1
1	0

(b)

x_1	y
0	0
1	0

(c)

x_1	y
0	1
1	1

(d)

Table 1: The four possible functions relating a dichotomous predictor variable x_1 to a dichotomous criterion variable y.

coding, for both x_1 and y, is essentially arbitrary—that is, you could equally well have coded the presence of attribute x_1 as 0 and its absence as 1—your learning machine should be equally capable of discovering a relationship such as that shown in Table 1(b). Similarly, had your machine uncovered the relationship shown in Table 1(c)—which shows that x_1 cannot be used to discriminate smuggles from non-smugglers—then it would, or *should*, have no difficulty in discovering the (non-predictive) relationship shown in Table 1(d).

As a result of the essential isomorphism of Tables 1(a) $\Leftrightarrow$ 1(b) and Tables 1(c) $\Leftrightarrow$ 1(d), the four functional relationships between x_1 and y that are illustrated in Tables 1(a) to 1(d) can be reduced to just two.

2 Progressing to 2-bits

Imagine now that rather than asking only whether an individual visited the Golden Triangle, you also ask whether they have syringes in their luggage. If they do, then you code variable x_2 as 1, otherwise as 0. Again, you submit your cases to your learning machine. What relationships might it discover? Since there are two dichotomous ‘predictor’ variables and one outcome variable, Table 2 must represent a complete tabulation of the possibilities.

Again, we can ask which of the 16 subtables are isomorphic under the rewriting rules described earlier, but there is now an additional consideration. Specifically, the decision to code visits to the Golden Triangle as variable x_1 and the presence of syringes in the luggage as variable x_2 is arbitrary. We could equally well exchange the variable names. As a consequence, Tables 2(d) and 2(f), for example, should be considered isomorphic, as can many of the other tables. A complete list of isomorphic tables is shown in Table 3.

3 Rules for isomorphism

The effect of the rewriting and reordering rules presented in the preceding sections, *functions* (since that is what each sub-table represents) are isomorphic if—

- Rewriting one or more variables x_i as $\neg x_i$ (i.e., changing $0 \rightarrow 1$ and $1 \rightarrow 0$),
- Rewriting variable y as $\neg y$ (i.e., changing $0 \rightarrow 1$ and $1 \rightarrow 0$), or
- Permuting the order of variables (i.e., reordering the columns of predictor variables)

would lead to the functions being identical.

x_2	x_1	y
0	0	0
0	1	0
1	0	0
1	1	0

(a)

x_2	x_1	y
0	0	0
0	1	0
1	0	0
1	1	1

(b)

x_2	x_1	y
0	0	0
0	1	0
1	0	1
1	1	0

(c)

x_2	x_1	y
0	0	0
0	1	0
1	0	1
1	1	1

(d)

x_2	x_1	y
0	0	0
0	1	1
1	0	0
1	1	0

(e)

x_2	x_1	y
0	0	0
0	1	1
1	0	0
1	1	1

(f)

x_2	x_1	y
0	0	0
0	1	1
1	0	1
1	1	0

(g)

x_2	x_1	y
0	0	0
0	1	1
1	0	1
1	1	1

(h)

x_2	x_1	y
0	0	1
0	1	0
1	0	0
1	1	0

(i)

x_2	x_1	y
0	0	1
0	1	0
1	0	0
1	1	1

(j)

x_2	x_1	y
0	0	1
0	1	0
1	0	1
1	1	0

(k)

x_2	x_1	y
0	0	1
0	1	0
1	0	1
1	1	1

(l)

x_2	x_1	y
0	0	1
0	1	1
1	0	0
1	1	0

(m)

x_2	x_1	y
0	0	1
0	1	1
1	0	0
1	1	1

(n)

x_2	x_1	y
0	0	1
0	1	1
1	0	1
1	1	0

(o)

x_2	x_1	y
0	0	1
0	1	1
1	0	1
1	1	1

(p)

Table 2: The 16 possible functions relating dichotomous predictor variables x_2 and x_1 to a dichotomous criterion variable, y.

Function shown in ...	is isomorphic with that shown in ...	by ...
Table 2(c)	Table 2(b)	$x_2 \leftrightarrow x_1$
Table 2(e)	Table 2(b)	$x_2 \rightarrow \neg x_2$
Table 2(f)	Table 2(d)	$x_2 \leftrightarrow x_1$
Table 2(h)	Table 2(b)	$x_2 \rightarrow \neg x_2, x_1 \rightarrow \neg x_1, y \rightarrow \neg y$
Table 2(i)	Table 2(b)	$x_2 \rightarrow \neg x_2, x_1 \rightarrow \neg x_1$
Table 2(j)	Table 2(g)	$y \rightarrow \neg y$
Table 2(k)	Table 2(d)	$x_2 \leftrightarrow x_1, y \rightarrow \neg y$
Table 2(l)	Table 2(b)	$x_2 \rightarrow \neg x_2, y \rightarrow \neg y$
Table 2(m)	Table 2(d)	$y \rightarrow \neg y$
Table 2(n)	Table 2(b)	$x_2 \rightarrow \neg x_2, y \rightarrow \neg y$
Table 2(o)	Table 2(b)	$y \rightarrow \neg y$
Table 2(p)	Table 2(a)	$y \rightarrow \neg y$

Table 3: The number of n-bit functions under the isomorphisim for $n \in \{0, 1, 2, 3, 4\}$.

Naming

Because the investigation of the isomorphism described in this paper was originally motivated by the development of a screening tool for identifying drug smugglers, we refer to the isomorphism as the *Smugglers isomorphism*.

4 How many n-bit functions are there?

We used *MATHEMATICA* to produce the 2^{2^n} functions of n-bits, for $n \in \{1, 2, 3, 4\}$ and to reduce them to a base set. We obtain the counts as shown in Table 4. The table shows clearly that if one wishes exhaustively to test one's learning machine, the number of functions against which the machine must be tested can be very much reduced from the number which is implied by column 2 of the table.

n	2^{2^n}	Non-isomorphic forms
0	2	1
1	4	2
2	16	4
3	256	14
4	65536	247

Table 4: The number of n-bit functions under the Smugglers isomorphism, for $n \in \{0, 1, 2, 3, 4\}$.