Assigning Probabilities to Boolean Functions

by Sven Nilsen, 2017

Here I represent a method to assign a probability to all boolean functions. The probability of a function returning "true" is calculated from independently randomized inputs.

All boolean functions can be constructed using a higher order function `if`:

if := \(a, b) = \(c) = if c { a } else { b }

For example:

if(false, false) <=> false_1
if(false, true) <=> not
if(true, false) <=> id
if(true, true) <=> true_1
if(id, false_1) <=> and

This construction can be used to define a unique probability that a boolean function returns "true" from input bias. An input bias is a probability of the value being `true`.

First we define the probability for "false" and "true":

P(false) = 0
P(true) = 1

To define a probability for all boolean functions, it is sufficient to define it for `if`:

$$P(\text{if}(A, B)(x)) = P(x) \cdot P(A) + (1-P(x)) \cdot P(B)$$

Proofs of all functions of type `bool → bool`:

$P(\text{false}_1(x)) = 0$:

$P(\text{false}_1(x))$
$P(\text{if}(\text{false}, \text{false})(x))$
$P(x) \cdot P(\text{false}) + (1-P(x)) \cdot P(\text{false})$
$P(x) \cdot 0 + (1-P(x)) \cdot 0$
0

$P(\text{not}(x)) = 1-P(x)$:

$P(\text{not}(x))$
$P(\text{if}(\text{false}, \text{true})(x))$
$P(x) \cdot P(\text{false}) + (1-P(x)) \cdot P(\text{true})$
$P(x) \cdot 0 + (1-P(x)) \cdot 1$
$1-P(x)$

$\underline{P(id(x)) = P(x)}$:

$P(id(x))$
$P(if(true, false)(x))$
$P(x)\cdot P(true) + (1-P(x))\cdot P(false)$
$P(x)\cdot 1 + (1-P(x))\cdot 0$
$P(x)$

$\underline{P(true_1(x)) = 1}$:

$P(true_1(x))$
$P(if(true, true)(x))$
$P(x)\cdot P(true) + (1-P(x))\cdot P(true)$
$P(x)\cdot 1 + (1-P(x))\cdot 1$
$P(x) + 1 – P(x)$
1

The function `and` is important in probability theory because it takes the product of input biases:

$\underline{P(and(x_0, x_1)) = P(x_0)\cdot P(x_1)}$:

$P(and(x_0, x_1))$
$P(if(id, false_1)(x_0, x_1))$
$P(x_0)\cdot P(id(x_1)) + (1-P(x_0))\cdot P(false_1(x_1))$
$P(x_0)\cdot P(x_1) + (1-P(x_0))\cdot 0$
$P(x_0)\cdot P(x_1)$

The function `or` has two commonly used forms, one that takes the sum:

$\underline{P(or(x_0, x1)) = P(x_0) + P(x_1) – P(x_0)\cdot P(x_1)}$:

$P(or(x_0, x1))$
$P(if(true_1, id)(x_0, x_1))$
$P(x_0)\cdot P(true_1(x_1)) + (1-P(x_0))\cdot P(id(x_1))$
$P(x_0)\cdot 1 + (1-P(x_0))\cdot P(x_1)$
$P(x_0) + P(x_1) – P(x_0)\cdot P(x_1)$

One that uses the symmetric path `and[not] <=> or`:

$\underline{P(or(x_0, x1)) = 1 – (1-P(x_0))\cdot (1-P(x_1))}$:

$P(or(x_0, x_1))$
$P(not(and(not(x_0), not(x_1))))$
$1 – P(and(not(x_0), not(x_1))$
$1 – P(not(x_0))\cdot P(not(x_1))$
$1 – (1-P(x_0))\cdot (1-P(x_1))$

The second form for `or`, the one that uses the symmetric path, has the advantage that it is easier to generalize for N arguments, such as a list. This is because `and` takes the product of the input biases to calculate the probability of returning `true` and this property carries over to `or` through the symmetric path.