

# Artificial Intelligence II

## Why multiplication of factors works

Sean B. Holden  
February 2010

### 1 Introduction

These notes provide a simple explanation of why the multiplication of factors in the manner suggested works, and why it avoids duplicating the computation of sub-expressions.

#### 1.1 Why it works

Let's drop any reference to probabilities for the moment and just look at general summations involving functions. Say you have three finite sets  $X = \{x_1, \dots, x_p\}$ ,  $Y = \{y_1, \dots, y_q\}$  and  $Z = \{z_1, \dots, z_r\}$ , and you want to compute a summation like

$$f(x, y) = \sum_{z \in Z} g(x, y, z)h(y, z) \tag{1}$$

for values  $x \in X$  and  $y \in Y$ . The sum will look like

$$f(x, y) = g(x, y, z_1)h(y, z_1) + \dots + g(x, y, z_r)h(y, z_r)$$

In other words, the products you need to compute are the ones *for which values of  $z$  coincide*. This, in a nutshell, is what the process of combining factors achieves. In this example, we would write the factors in the sum as

$x$	$y$	$z$	$F_x(x, y, z)$
$x_1$	$y_1$	$z_1$	$g(x_1, y_1, z_1)$
$x_1$	$y_1$	$z_2$	$g(x_1, y_1, z_2)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$x_p$	$y_q$	$z_r$	$g(x_p, y_q, z_r)$

and

$y$	$z$	$F_y(y, z)$
$y_1$	$z_1$	$h(y_1, z_1)$
$y_1$	$z_2$	$h(y_1, z_2)$
$\vdots$	$\vdots$	$\vdots$
$y_q$	$z_r$	$h(y_q, z_r)$

When the factors are multiplied we match up and multiply the table entries for which variables common to both have matching values. So for example

$x$	$y$	$z$	$F_{x,y}(x, y, z)$
$x_1$	$y_1$	$z_1$	$g(x_1, y_1, z_1)h(y_1, z_1)$
$x_1$	$y_1$	$z_2$	$g(x_1, y_1, z_2)h(y_1, z_2)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$x_p$	$y_q$	$z_r$	$g(x_p, y_q, z_r)h(y_q, z_r)$

In order to deal with the summation to form the factor  $F_{x,y}(x, y)$  we now form sums of the entries in  $F_{x,y}(x, y, z)$  over all values for  $z$ , so

$x$	$y$	$F_{x,y}(x, y)$
$x_1$	$y_1$	$\sum_{z \in Z} F_{x,y}(x_1, y_1, z)$
$x_1$	$y_2$	$\sum_{z \in Z} F_{x,y}(x_1, y_2, z)$
$\vdots$	$\vdots$	$\vdots$
$x_p$	$y_q$	$\sum_{z \in Z} F_{x,y}(x_p, y_q, z)$

Expanding out one of these summations results in something like

$$F_{x,y}(x_1, y_2) = \sum_{z \in Z} g(x_1, y_2, z)h(y_2, z) \quad (2)$$

Comparing equations 1 and 2 we see that the entries in the factor  $F_{x,y}(x, y)$  are just the values of the summation for each possible pair of values  $x$  and  $y$ .

## 1.2 The relationship to probabilities

So: the tabular process using factors is just a way of keeping track of the values needed to compute the sum. In the probabilistic inference algorithm the functions  $f$ ,  $g$ ,  $h$  and so on are all just (conditional) probability distributions, and because we're dealing the the decomposition

$$\Pr(X_1, \dots, X_n) = \prod_{i=1}^n \Pr(X_i | \text{parents}(X_i))$$

on a directed, acyclic graph we start off with a factor for each RV, and each time we get to a summation we sum out the corresponding variable. (The above example does not have this structure, which is why the summing out notation  $F_{x,y,\bar{z}}$  does not appear, but the process is identical.)

## 1.3 Why it avoids duplication

Reverting now to probabilities, say we have a Bayes network that represents the decomposition

$$\Pr(X, Y_1, Y_2, E_1, E_2) = \Pr(E_1 | Y_1, Y_2) \Pr(E_2 | Y_2) \Pr(Y_2 | X) \Pr(X) \Pr(Y_1)$$

(Exercise: draw it.) Now we attempt to compute the inference

$$\Pr(X | e_1, e_2) = \frac{1}{Z} \Pr(X) \sum_{y_1 \in Y_1} \Pr(Y_1) \sum_{y_2 \in Y_2} \Pr(Y_2 | X) \Pr(e_1 | Y_1, Y_2) \Pr(e_2 | Y_2)$$

The repetition of computations arises in summations like this because—handled in the naive way using recursive depth-first evaluation—the summation

$$\sum_{y_2 \in Y_2} \Pr(Y_2 | X) \Pr(e_1 | Y_1, Y_2) \Pr(e_2 | Y_2)$$

will involve computing the product  $\Pr(Y_2 | X) \Pr(e_2 | Y_2)$  for each value of  $Y_1$ . This problem will repeat itself for each value of  $X$ . By computing and storing each of the products needed only once the method based on factors avoids this.