Potential operations in the `gRain` package

Søren Højsgaard

May 1, 2008

Contents

1 Potentials and operations on these

Consider a set $\Delta = \{\delta_1, \dots, \delta_R\}$ of discrete variables where δ_r has a finite set I_r of levels. Let $|I_r|$ denote the number of levels of δ_r and let $i_r \in I_r$ denote a value of δ_r. A configuration of the variables in Δ is then $i = (i_1, \dots, i_R) \in I_1 \times \ldots \times I_R$. The total number of configurations is then $|\Delta| = \prod_r |I_r|$. Let U be a non–empty subsets of Δ with configurations I_U and let i_U denote a specific configuration.

A potential T_U defined on I_U is a non–negative function, i.e. $T_U(i_U) \geq 0$ for all $i_U \in I_U$.

Let U and V be non–empty subsets of Δ with configurations I_U and I_V and let T^1_U and T^2_V be corresponding potentials.

The product (quotient) of T^1_U and T^2_V are potentials defined on $U \cup V$ given by

$$T_{U\cup V} := T^1_U \times T^2_V \text{ and } T_{U\cup V} := T^1_U / T^2_V$$

respectively, with the convention that $0/0 = 0$. If $V \subset U$ is non–empty[1] then marginalization of T^1_U onto V is defined as

$$T^1_V := \sum_{U\setminus V} T^1_U$$

[1] Marginalization onto an empty set is not implemented.

1.1 Implementation of potentials

Potentials are represented by `ptab` objects which are defined as part of the `gRain` package. `ptab` objects are essentially arrays, and the only reason for not simply working with arrays implementing a special class is a pure technicality: Two–dimensional arrays are (correctly) in some respects regarded as matrices while one–dimensional arrays are (correctly) in some respects regarded as vectors. For our purpose we need a class of objects which are regarded as being of the same type irrespectively of their dimension. However we may for all practical purposes think of `ptab` objects as arrays.

1.2 Examples

`ptab` objects can be created as:

```
> yn <- c("y", "n")

[1] "y" "n"

> a.1 <- ptab("asia", list(yn), values = c(1, 99))

asia
 y  n
 1 99

> t.a.1 <- ptab(c("tub", "asia"), list(yn, yn), values = c(5, 95, 1, 99))

   asia
tub  y  n
  y  5  1
  n 95 99
```

Tables can be normalized in two ways: Either the values are normalized over all configurations to sum to one as

```
> a.2 <- ptab("asia", list(yn), values = c(1, 99), normalize = "all")

asia
   y    n
0.01 0.99
```

Alternatively normalization can be over the first variable for *each* configuration of all other variables as

```
> t.a.2 <- ptab(c("tub", "asia"), list(yn, yn), values = c(5, 95, 1, 99), normalize = "first")

   asia
tub    y    n
  y 0.05 0.01
  n 0.95 0.99
```

1.3 Operations on potentials

Multiplication and division of potentials is implented as follows. Consider multiplication of ϕ_U and ψ_V.

The vectors, say T_U and T_V, containing the values of the potentials are given a dimension attribute, i.e. are turned into arrays.

Assume first that $V \subset U$. Then we reorder the elements of T_U to match with those of T, symbolically as $(V, U \setminus V)$ so that we have tables T_V into $T_{V,U\setminus V}$ accordingly. This operation is fast with the `aperm()` function which is implemented in C. We can then form the product $T_{V,U\setminus V}T_V$ directly because the elements of T_V are recycled to match the length of $T_{V,U\setminus V}$. If V is not a subset of U then we expand the domain of T_U into $T_{V,U\setminus V}$ by first permuting the array with `aperm()` and then repeating the entries a suitable number of times and then carry out the multiplications as above.

Marginalization is similarly based on using `apply()` where summation is over a specific set of dimensions.

1.4 Examples

Hence we can calculate the joint, the marginal and the conditional distributions as

```
> ta.1 <- arrayop(t.a.1, a.1, op = "*")
> ta.1

    tub
asia  y    n
   y  5   95
   n 99 9801

> arraymarg(ta.1, "tub")

tub
   y    n
 104 9896

> arrayop(ta.1, arraymarg(ta.1, "tub"), op = "/")

   asia
tub           y         n
  y 0.048076923 0.9519231
  n 0.009599838 0.9904002
```

The `ptab` function takes a `smooth` argument which by default is 0. A non-zero value of `smooth` implies that zeros in `values` are replaced by the value of smooth – before any normalization is made, e.g.

```
> ptab(c("tub", "asia"), list(c("y", "n"), c("y", "n")), values = c(0, 95, 0,
+     99), normalize = "first", smooth = 1)


   asia
tub          y    n
  y 0.01041667 0.01
  n 0.98958333 0.99
```

It is possible to take out a sub–array defined by specific dimensions being at specific levels. This corresponds finding a specific slice of a multidimensional array: To find the 1–dimensional array defined by asia (variable 1) being "no" (at level 2) do:

```
> ta.1


    tub
asia  y    n
   y  5   95
   n 99 9801


> subarray(ta.1, margin = 1, index = 2)


tub
   y    n
  99 9801
```

1.5 Coercion

Coercion to to `ptab` is done by `as.ptab`. For example:

```
> v <- 1:4
> as.ptab(v)


V1
V11 V12 V13 V14
  1   2   3   4


> names(v) <- c("a1", "a2", "a3", "a4")
> as.ptab(v)


V1
V11 V12 V13 V14
  1   2   3   4
```

```
> v <- array(1:4, c(4))
> as.ptab(v)


V1
V11 V12 V13 V14
  1   2   3   4


> v <- array(1:4, c(4), dimnames = list(a = c("a1", "a2", "a3", "a4")))
> as.ptab(v)


a
a1 a2 a3 a4
 1  2  3  4
```

```
> v <- array(1:4, c(2, 2))
> as.ptab(v)


     V2
V1    V21 V22
  V11   1   3
  V12   2   4


> v <- array(1:4, c(2, 2), dimnames = list(a = c("a1", "a2"), b = c("b1", "b2")))
> as.ptab(v)


    b
a    b1 b2
  a1  1  3
  a2  2  4
```

```
> v <- array(1:8, c(2, 2, 2))
> as.ptab(v)

, , V3 = V31

     V2
V1    V21 V22
  V11   1   3
  V12   2   4

, , V3 = V32

     V2
V1    V21 V22
  V11   5   7
  V12   6   8

> v <- array(1:4, c(2, 2, 2), dimnames = list(a = c("a1", "a2"), b = c("b1", "b2"),
+     c = c("c1", "c2")))
> as.ptab(v)

, , c = c1

    b
a    b1 b2
  a1  1  3
  a2  2  4

, , c = c2

    b
a    b1 b2
  a1  1  3
  a2  2  4
```