中山大学数据科学与计算机学院
计算机科学与技术专业-人工智能
本科生实验报告

（2018-2019 学年秋季学期）

学　号：16337113
姓　名：劳马东
教学班级：教务 2 班
专　业：超算

一. 实验题目

实现变量消元算法的 $inference, multiply, sumout$ 和 $restrict$ 函数，求解：

P(Alarm)

P(J && ~M)

P(A | J && ~M)

P(B | A)

P(B | J && ~M)

P(J && ~M | ~B)

二. 实验内容

（一）因子（factors）

在变量消元过程中，经常会遇到像这样的式子——$\sum_A Pr(A)\sum_B Pr(B)Pr(C|A,B)$，它实际上是求 $Pr(A,B,C)$。我们将先验分布 $Pr(A)$、$Pr(B)$ 和条件分布 $Pr(C|A,B)$ 抽象为因子，统一表示为 $f(A)$、$f(B)$ 和 $f(C,A,B)$。因子 $f(V_1,V_2,...,V_n)$ 是随机变量集合 $\{V_1,V_2,...,V_n\}$ 到非负实数域的映射，每个随机变量实例化后的 $f(v_1,v_2,...,v_n)$ 是对应的条件概率或联合概率。例如，$f(a)=Pr(A=a)$，$f(a,b,c)=Pr(c|a,b)$。

个人认为，引入因子的原因，一方面是便于书写，不必考虑随机变量之间的关系（如哪些变量是前提）；另一方面是更好地重用因子，因为在推理过程中有很多重复计算，如果保存推理过程中得到的因子，结合因子涉及的随机变量，能很快发现哪些因子可以重用，减少不必要计算。

（二）因子的基本操作

1、 对某个变量 V 求和（sum-out）

假设存在因子 $f(V_1,V_2,...,V_n)$，它要在随机变量 V_1 上求和，从而产生一个新的因子，过程如下：

（1） $\forall v\in V_1$，将除 V_1 外其余变量取值相同的 $f(v,V_2,...,V_n)$ 相加；

（2） 将 V_1 从 f 中删除，得到一个新的因子 $g(V_2,...,V_n)$，其中 $g(v_2,...,v_n)=\sum_{v\in V_1} f(v,v_2,...,v_n)$。

算法 1 因子求和

---

function $sumout(f,i)$

输入: 因子 f，f 要消除的变量的下标 i

输出: 求和后的新因子 g

　$V_1,V_2,...,V_n \leftarrow f.variables$

　$g \leftarrow$ new factor with variables $\{V_1,...,V_{i-1},V_{i+1},...,V_n\}$

　for each tuple $(v_1,...,v_{i-1},v_{i+1},...,v_n)\ \in\ g$ **do**

　　$g[(v_1,...,v_{i-1},v_{i+1},...,v_n)] \leftarrow 0$

　end for

　for each tuple $(v_1,v_2,...,v_n)\ \in\ f$ **do**

　　$g[(v_1,...,v_{i-1},v_{i+1},...,v_n)] \leftarrow g[(v_1,...,v_{i-1},v_{i+1},...,v_n)] + f[(v_1,v_2,...,v_n)]$

　end for

　return g

end function

2、 因子相乘（factor multiplication）

假设因子 $f(A,B)$ 与 $g(B,C)$ 具有相同的变量 B，则 f 与 g 的乘积，记为 $h(A,B,C)=f(A,B)\times g(B,C)$，是它们的自然连接，概率是对应元组概率的乘积。例如，对于表1中的例子，给出 $f(A,B)$ 和 $g(B,C)$ 各种取值的概率，ab 与 bc 自然连接产生 abc，它的概率是 $f(a,b)\times g(b,c)=0.9\times 0.7=0.63$，$ab$ 与 $\sim bc$ 自然连接时由于 $b\neq\sim b$，不产生结果。同理可以得出其余自然连接的结果。

f(A, B)		g(B, C)		h(A, B, C)			
ab	0.9	bc	0.7	abc	0.63	ab∼c	0.27
a∼b	0.1	b∼c	0.3	a∼bc	0.08	a∼b∼c	0.02
∼ab	0.4	∼bc	0.8	∼abc	0.28	∼ab∼c	0.12
∼a∼b	0.6	∼b∼c	0.2	∼a∼bc	0.48	∼a∼b∼c	0.27

表 1. 因子相乘示例

算法 2 因子相乘

function $multiply(f,g)$

输入: 因子 f 和 g

输出: 相乘得到的新因子 h

  $common \leftarrow f.variables \cap g.variables$

  $h \leftarrow$ new factor with variables $f.variables \cup g.variables$

  for each tuple $(x_1,x_2,...,x_m)\in f$ **do**

    for each tuple $(y_1,y_2,...,y_n)\in g$ **do**

      if $(x_1,x_2,...,x_m)$ is the same as $(y_1,y_2,...,y_n)$ on *common* **then**

        $h[(x_1,x_2,...,x_m)\cup(y_1,y_2,...,y_m)] \leftarrow f[(x_1,x_2,...,x_m)]\times g[(y_1,y_2,...,y_n)]$

      end if

    end for

  end for

  return h

end function

（三）变量消元过程

假设给定一个贝叶斯网络，CPT 表为 F，查询变量为 Q，证据变量 E 带观察值 e，剩余未消除的变量集合为 Z，计算 $Pr(Q|E)$。

1、 证据变量赋值（Restriction）

对于 F 中的所有包含变量 E 的因子 f，将它上 $E\neq e$ 的元组删除，只保留那些 $E==e$ 的元组，最后产生一个新的因子 g。将 g 加入 F，将 f 从 F 中删除。如果 f 只包含 E 这一个变量，g 将是一个“常量”因子，可不放入 F 中。

2、 按顺序消除变量

对于给定消元顺序中的每个变量 $Z_j\in Z$，用以下步骤消除它：

- 假设因子 $f_1,f_2,...,f_k$ 包含变量 Z_j；
- 计算新因子 $g_j=\sum_{Z_j}\prod_{i=1}^{k} f_i$；
- 将 $f_1,f_2,...,f_k$ 从 F 中删除，并加入新的因子 g_j。

3、 含查询变量的因子相乘

消除完成后，Z 中剩下的因子都是只和 Q 相关的因子，记为 $f_1(Q), f_2(Q), ..., f_k(Q)$，$f(Q) = \prod_{i=1}^{k} f_i(Q)$。

4、 归一化

$f(Q)$ 上的概率相加的和不一定是 1，尤其是在求条件概率时概率和通常大于 1，因此需要归一化，即 $\forall t \in f, f[t] = \alpha f[t]$，其中 $\alpha = \frac{1}{\sum_{t \in f} f[t]}$。

现在考虑一个具体的例子，如图1。在图右边的贝叶斯网络中，变量 C 的父节点是 A 和 B，变量 D 的父节点是 C，因此 CPT 中的因子就是 $f_1(A)$、$f_2(B)$、$f_3(A,B,C)$、$f_4(C,D)$。现在，在观察到证据变量 $D = d$ 的前提下，计算查询变量 A 的概率，即 $Pr(A|D = d)$，变量消元的顺序是 C、B。

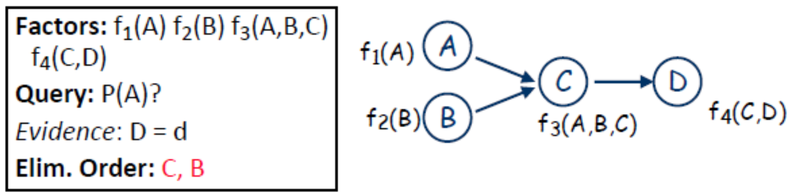

图 1. 变量消元例子

第一步，证据变量赋值。证据变量是 D，因子 $f_4(C,D)$ 与 D 相关，于是限制 f_4 上 D 的值为 d，产生新的因子 $f_5(C)$；

第二步，按给定顺序消除变量。首先消除 C，计算并添加因子 $f_6(A,B) = \sum_C f_5(C) f_3(A,B,C)$，删除因子 f_3 和 f_5；然后消除 B，计算并添加因子 $f_7(A) = \sum_B f_2(B) f_6(A,B)$，删除 f_2 和 f_6。

第三部，现在剩下的因子是 $f_1(A)$ 和 $f_7(A)$，它们都只和 A 相关，计算 $f_1(A) \times f_7(A)$。

最后，归一化。$Pr(A|D = d) = \alpha f_1(A) \times f_7(A)$，其中 $\alpha = \frac{1}{\sum_A f_1(A) \times f_7(A)}$。

三. 关键代码

1、 因子求和

首先是将要求和的变量 *variable* 从变量列表中删除，得到新的变量列表 *new_var_list*，*var_index* 是 *variable* 在旧变量列表中的下标。

然后，遍历 *cpt* 中的每一个键值对，键是变量取值，值是对应的概率。新的键 *new_k* 由旧键删除第 *var_index* 个元素产生，新的值时相同 *new_k* 对应概率的累加和。

```
def sum_out(self, variable):
  # 找到variable在var_list中的下标，并将其删除产生新的变量列表
  var_index, new_var_list = self.without(variable)
  new_cpt = dict()    # 利用字典统计概率的累加和
  for k, v in self.cpt.items():
    new_k = k[:var_index] + k[var_index + 1:]   # 用分片实现字符串删除第var_index个元素
    new_pro = new_cpt.get(new_k, 0) + v
    new_cpt[new_k] = new_pro
  # 尝试一个新的因子，变量列表是new_var_list，概率表是new_cpt
  new_node = Node('f' + str(new_var_list), new_var_list)
  new_node.set_cpt(new_cpt)
  return new_node
```

2、 因子相乘

如上文所述，因子相乘其实是自然连接的过程。因此，第一步是找到两个因子相同的变量集合，这借助于 $interaction$ 函数，它返回两个列表共有的元素列表。然后，使用一个二重循环变量两个因子 cpt 表上的每一项，用 $join$ 函数在 $common$ 上连接两个项，返回 $t1 \cup t2$，如果这两个项无法连接将返回 $None$。

```
def multiply(self, factor):
  new_cpt = dict()
  common = []
  # 求两个因子的交集，并在common中添加这些交集变量对应的下标
  for c in Util.interaction(self.var_list, factor.var_list):
    common.append((self.get_var_index(c), factor.get_var_index(c)))
  for t1, v1 in self.cpt.items():
    for t2, v2 in factor.cpt.items():
      k = Util.join(t1, t2, common) # 按common上的属性，自然连接t1和t2
      if k is not None:
        new_cpt[k] = v1 * v2
  new_var_list = self.var_list + factor.var_list
  for i, _ in common:   # 求并集，删除self.var_list + factor.var_list中重复的元素
    new_var_list.pop(i)
  new_node = Node('f' + str(new_var_list), new_var_list)
  new_node.set_cpt(new_cpt)
  return new_node
```

3、 按顺序消除变量

在消除的过程中，将因子列表 $factor\_list$ 看做一个队列，每次取队头元素，如果它包含要消除的变量 var，就将它和同样具有 var 变量的因子相乘，否则重新将它放入队列。

```
for var in ordered_list_of_hidden_variables:
  new_f = None
  for i in range(len(factor_list)):
    factor = factor_list.pop(0)   # 移除队头的因子
    if var in factor.var_list:
      # 将所有含var变量的因子相乘
      if new_f is None:
        new_f = factor
      else:
        new_f = new_f.multiply(factor)
    else:   # 因子不包含var变量，不能相乘，重新放入队列
      factor_list.append(factor)
  if new_f is not None:   # 相乘后是求和操作
    new_f = new_f.sum_out(var)
    factor_list.append(new_f)
```

4、 归一化

```
def normalize(cpt, query_variables):
  n = len(query_variables)  # 默认因子的前n个是非条件
  totals = dict()
  for k, v in cpt.items():
    totals[k[n:]] = totals.get(k[n:], 0) + v
  for k in cpt:
    cpt[k] /= totals[k[n:]]
```

四. 实验结果

	证据变量	查询变量	消元顺序	key	概率
P(A)	-	'A'	'B'、'E'、'J'、'M'	'1'	0.002516442
P(J && ∼M)	-	'J'、'M'	'B'、'E'、'A'	'10'	0.050054875461
P(A | J && ∼M)	'J': '1', 'M': '0'	'A'	'B'、'E'	'1'	0.013573889331307633
P(B | A)	'A': '1'	'B'	'E'、'J'、'M'	'1'	0.373551228281836
P(B | J && ∼M)	'J': '1', 'M': '0'	'B'	'E'、'A'	'1'	0.0051298581334013015
P(J && ∼M | ∼B)	'B': '0'	'J'、'M'	'E'、'A'	'10'	0.049847949