Aprendizaje Automático

Modelos Gráficos Probabilísticos

Vicente Bosch, Enrique Vidal, Francisco Casacuberta
Departamento de Sistemas Informáticos y Computación
Universitat Politècnica de València

Curso 2019-2020

1. Introducción

El objetivo de esta práctica es experimentar con Modelos Gráficos Probabilísticos, también conocidos como Redes Bayesianas. Para ello utilizaremos el *toolkit* "BNT" para `matlab`, desarrollado por Kevin Murphy[1]. Este *toolkit* se puede descargar de:

`https://code.google.com/p/bnt` – y también: `https://github.com/bayesnet/bnt`

Un manual de uso bastante completo, con ejemplos que incluyen algunos que se reproducen en este boletín puede encontrase en:

`http://bayesnet.github.io/bnt/docs/usage.html`

Puede instalarse fácilmente siguiendo las instrucciones que se encuentran en esta página web. Una instalación ya preparada para su uso en esta práctica, está disponible en:

`~/asigDSIC/ETSINF/apr/bayes/BNT`

La implementación de BNT se basa en el uso de matrices multidimensionales, que no están completamente soportadas por `octave`. Por tanto, a diferencia de las anteriores prácticas, en esta usaremos `matlab`, cuya sintaxis y modo de operación son muy similares (o casi idénticos) a `octave`. Con la invocación `matlab` sin argumentos se obtiene una interfaz de usuario gráfica completa. No obstante, a los efectos de esta práctica, basta con utilizar la interfaz textual, que es más simple y ligera y se obtiene mediante:

```
matlab -nodesktop -nosplash
```

Una vez inicializado el entorno, deberemos de añadir al `path` de `matlab` el directorio donde se encuentra la instalación mediante las siguientes instrucciones.

```
addpath('~/asigDSIC/ETSINF/apr/bayes/BNT')
addpath(genpathKPM('~/asigDSIC/ETSINF/apr/bayes/BNT'))
```

De esta forma se evita tener que copiar la instalación completa de BNT (que es muy grande) a nuestro `home`. *Importante: hay que realizar este paso cada vez que se arranque un nuevo entorno de trabajo.*

[1] Kevin Murphy. "The Bayes net toolbox for MATLAB." Computing science and statistics 33.2 (2001): 1024-1034. `http://people.cs.ubc.ca/~murphyk/Papers/bnt.pdf`

2. Ejemplo simple de uso de BNT: *"Sprinkler"*

Para introducir la funcionalidad básica de BNT en matlab nos basaremos en un ejemplo simple adaptado del libro de Russell y Norvig, "Artificial Intelligence: a Modern Approach", Prentice Hall, 1995, p454:

C	$P(C)$	$P(S \mid C), S =$ false	$P(S \mid C), S =$ true	$P(R \mid C), R =$ false	$P(R \mid C), R =$ true
false	0.5	0.5	0.5	0.8	0.2
true	0.5	0.9	0.1	0.2	0.8

S	R	$P(W \mid S,R), W =$ false	$P(W \mid S,R), W =$ true
false	false	1.0	0.0
true	false	0.1	0.9
false	true	0.1	0.9
true	true	0.01	0.99

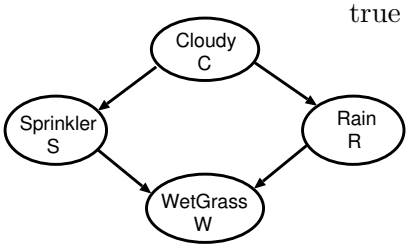

Figura 1: Red bayesiana simple para el ejemplo *"Sprinkler"*

Grafo acíclico dirigido

Empezaremos por definir el grafo acíclico dirigido (DAG) del ejemplo mediante una matriz de adyacencia. Para definir dicha matriz hemos numerado topológicamente los nodos; es decir, los antecesores han de estar antes que los descendientes (es imprescindible seguir este orden):

1. Nublado (Cloudy, C)
2. Aspersor (Sprinkler, S)
3. Llueve (Rain, R)
4. Césped húmedo (WetGrass, W)

La matriz de adyacencia del ejemplo *Sprinkler* se define en BNT como:

```
N = 4;  C = 1; S = 2; R = 3; W = 4;
grafo          = zeros(N, N);
grafo(C,[R S]) = 1;
grafo(R,W)     = 1;
grafo(S,W)     = 1;
```

El grafo definido por esta matriz de adyacencia puede visualizarse mediante `draw_graph(grafo);` aunque en algunas versiones de matlab esta invocación suele producir errores.

Además hay que especificar la *variable aleatoria* asociada a cada nodo; es decir, su tipo y tamaño. Si es discreta, el tamaño del nodo es el número total de posibles valores que puede tomar la variable. Por otra parte si es de tipo real/continuo, puede ser un vector y el tamaño es la dimensión de dicho vector. En el ejemplo *Sprinkler* todos los nodos son binarios y por tanto discretos:

```
nodosDiscretos = 1:N;
tallaNodos     = 2*ones(1,N);
```

Si los nodos fueran de tamaños diferentes, se especificaría un tamaño distinto para cada nodo, por ejemplo, de la siguiente forma: `tallaNodos = [4 2 3 5];` Finalmente, se crea la red bayesiana uniendo el DAG y la especificación de los nodos:

```
redB = mk_bnet(grafo, tallaNodos,  'discrete', nodosDiscretos);
```

Conviene notar que, para usar cualquiera de los parámetros opcionales que permite una función, solo hay que añadir a la llamada el par formado por el *valor* y el *nombre* , como hemos visto para `mk_bnet`.

2.1. Probabilidades condicionales

Para completar la especificación de una red bayesiana, aparte de definir el DAG y los nodos correspondientes, hay que indicar las Distribuciones de Probabilidad Condicional (*Conditional Probability Distribution*, CPD) asociadas a cada nodo. Para un nodo cuya variable aleatoria asociada es Z, su CPD especifica la probabilidad condicional $P(Z \mid \text{pred})$, donde "pred" es el conjunto de variables asociadas a los nodos antecesores. La CPD más simple es una Tabla, que abreviaremos como TPC ("Tabla de Probabilidades Condicionales" – en BNT se le llama CTP). Esto es adecuado para nodos de naturaleza discreta, como es el caso en este ejemplo.

Es importante notar que los valores de una variable aleatoria discreta no están ordenados de ninguna forma en particular; es decir, son valores categóricos y no ordinales.

Las dimensiones de cada TPC deben de estar ordenadas según el orden definido en los nodos del DAG. Si un nodo no tiene antecesores, su TPC es un vector columna con la probabilidad para cada valor de ese nodo. Por ejemplo, la TPC para el nodo 4 (WetGrass, césped húmedo) de la figura1, con los nodos en el orden definido es:

S	R	W	$P(W \mid S, R)$
false	false	false	1.0
true	false	false	0.1
false	true	false	0.1
true	true	false	0.01
false	false	true	0.0
true	false	true	0.9
false	true	true	0.9
true	true	true	0.99

Para especificar la TPC de un nodo, en `matlab` podemos definir cada uno de los valores de la matriz uno a uno; por ejemplo:

```
TPC_W        = zeros(2, 2, 2);
TPC_W(1,1,1) = 1.0;
TPC_W(2,1,1) = 0.1;
TPC_W(1,2,1) = 0.1;
TPC_W(2,2,1) = 0.01;
...
```

Pero resulta más sencillo y eficaz utilizar el constructor de TPC proporcionado por `BNT`:

```
redB.CPD{W} = tabular_CPD(redB, W, [1.0 0.1 0.1 0.01 0.0 0.9 0.9 0.99]);
```

donde el orden de los elementos ha de ser el mismo que el de los valores de $P(W \mid S, R)$ en la tabla anterior. Y de forma similar para los demás nodos:

```
redB.CPD{C} = tabular_CPD(redB, C, [0.5 0.5]);
redB.CPD{S} = tabular_CPD(redB, S, [0.5 0.9 0.5 0.1]);
redB.CPD{R} = tabular_CPD(redB, R, [0.8 0.2 0.2 0.8]);
```

Esta RB representa la distribución: $P(C, S, R, W) = P(C)P(S \mid C)P(R \mid C)P(W \mid S, R)$, en la que todas las variables aleatorias toman valores en $\{\text{false} \equiv 1, \text{true} \equiv 2\}$.

2.2. Inferencia

Una vez creada la red bayesiana, podemos usarla para inferir cualquier distribución de probabilidad sobre las variables aleatorias definidas por la red. Hay diversos algoritmos de inferencia para redes bayesianas, con distintos compromisos entre velocidad, complejidad, generalidad y precisión. `BNT` ofrece varios de estos algoritmos, pero en esta práctica solo usaremos el motor de inferencia denominado *junction tree engine*:

```
motor = jtree_inf_engine(redB);
```

Otros motores de inferencias tienen constructores muy similares a este, pero pueden requerir parámetros adicionales. Una vez definida una instancia de un motor de inferencia, sea cual sea, se usan todos de la misma forma.

Distribuciones condicionales

Supongamos que queremos calcular la probabilidad de que el aspersor (S) esté parado, o en marcha, si la hierba está mojada (es decir, $W = \text{true} \equiv 2$). Formalmente hay que calcular $P(S = s \mid W = \text{true})$ para $s \in \{\text{false}, \text{true}\}$ o, en BNT, $P(S = s \mid W = 2)$ para $s \in \{1, 2\}$; concretamente:

$$P(S = s \mid W = 2) \;=\; \frac{\sum_{c\in\{1,2\}} \sum_{r\in\{1,2\}} P(C = c, S = s, R = r, W = 2)}{\sum_{c'\in\{1,2\}} \sum_{s'\in\{1,2\}} \sum_{r'\in\{1,2\}} P(C = c', S = s', R = r', W = 2)}$$

La evidencia es pues $W = 2$, lo que se especifica como sigue:

```
evidencia    = cell(1, N);
evidencia{W} = 2;
```

Usamos un *cell array* para poder acomodar diferentes tipos de evidencia o tamaño para cada nodo. `BNT` entiende el valor `[]` para denotar que un nodo no tiene evidencia. Una vez creada, se inserta la evidencia en un motor de inferencia de la siguiente forma:

```
[motor, logVerosim] = enter_evidence(motor, evidencia);
```

Esta función devuelve el motor con la evidencia "incorporada" y la log-verosimilitud correspondiente. Finalmente, podemos calcular $P(S = s \mid W = 2)$ para $s \in \{\text{false,true}\}$ como sigue:

```
m = marginal_nodes(motor, S);
m.T
ans =
      0.5702
      0.4298
```

El resultado, `m`, es una estructura cuyo campo `T` es un vector que contiene la distribución de probabilidad para el nodo especificado (S). Así pues, $P(S = 2 \mid W = 2) = 0.4298$. Si ahora, por ejemplo, se añade la evidencia de que está lloviendo (o sea, $R = \text{true} \equiv 2$), la nueva distribución se calcula como:

$$P(S{=}s \mid R{=}2, W{=}2) = \frac{\sum_{c\in\{1,2\}} P(C{=}c, S{=}s, R{=}2, W{=}2)}{\sum_{c'\in\{1,2\}} \sum_{s'\in\{1,2\}} P(C{=}c', S{=}s', R{=}2, W{=}2)}$$

Y en BNT:

```
evidencia{R}        = 2;
evidencia{W}        = 2;
[motor, logVerosim] = enter_evidence(motor, evidencia);
m                   = marginal_nodes(motor, S);
m.T
ans =
      0.8055
      0.1945
```

En este caso, $P(S = 2 \mid R = 2, W = 2) = 0.1945$, que es menor que en el caso anterior ya que la certeza de lluvia explica en buena medida que el césped esté húmedo, sin que sea "tan necesario" que aspersor esté en marcha.

Nodos observados

¿Que ocurre si se calcula la probabilidad condicional de un nodo observado; por ejemplo, $P(W \mid W = w)$? Obviamente, para todo w, $P(W = w \mid W = w) = 1$ y $P(W \neq w \mid W = w) = 0$. BNT trata los nodos discretos observados como si solo tuvieran un valor, con probabilidad 1.0. Esto puede verse en el siguiente ejemplo:

```
evidencia    = cell(1, N);
evidencia{W} = 2;
motor        = enter_evidence(motor, evidencia);
m            = marginal_nodes(motor, W);
m.T
ans =
    1.0000
```

Puede resultar un tanto confuso que solo se muestre una probabilidad, en vez de una para cada valor de la variable aleatoria asociada al nodo considerado. Si, por mayor claridad, se desea la respuesta completa hay que añadir un argumento adicional:

```
m = marginal_nodes(motor, W, 1);
m.T
ans =
         0
    1.0000
```

Así se obtienen de forma explícita las probabilidades de todos los valores; dos en este ejemplo: $P(W = 1 \mid W = 2) = 0$ y $P(W = 2 \mid W = 2) = 1$.

Distribuciones Conjuntas

En el siguiente ejemplo calcularemos la probabilidad conjunta de un grupo de nodos: `S,R,W` cuando no hay ningún nodo observado (es decir la evidencia está vacía):

$$P(S=s, R=r, W=w) \;=\; \sum_{c\in\{1,2\}} P(C=c, S=s, R=r, W=w)$$

Y en BNT:

```
evidencia            = cell(1,N);
[motor, logVerosimi] = enter_evidence(motor, evidencia);
m                    = marginal_nodes(motor, [S R W]);
```

donde ahora el campo `T` de `m` es un vector multi-dimensional (en este caso de tres dimensiones) que contiene la distribución de probabilidad conjunta para los nodos especificados; es decir, $T(s,r,w) \equiv P(S=s, R=r, W=w \mid \text{evidencia})$.

```
m.T
ans(:,:,1) =
    0.2900    0.0410
    0.0210    0.0009
ans(:,:,2) =
         0    0.3690
    0.1890    0.0891
```

Vemos que $P(S=1, R=1, W=2) = 0$. Esto tiene sentido puesto que es totalmente improbable que el césped esté húmedo ($W=2$) cuando no llueve ($R=1$) y el aspersor está parado ($S=1$). Si ahora se incluye la evidencia de lluvia ($R=2$):

$$P(S=s, R=r, W=w \mid R=2) = \frac{\sum_{c\in\{1,2\}} P(C=c, S=s, R=r, W=w)}{\sum_{c'\in\{1,2\}} \sum_{s'\in\{1,2\}} \sum_{w'\in\{1,2\}} P(C=c', S=s', R=2, W=w')}$$

Y en BNT:

```
evidencia{R} = 2;
[motor, ll] = enter_evidence(motor, evidencia);
m = marginal_nodes(motor, [S R W])
m =
    domain: [2 3 4]
         T: [2x1x2 double]
    ...
m.T
ans(:,:,1) =
    0.0820
    0.0018
ans(:,:,2) =
    0.7380
    0.1782
```

Las probabilidades $T(s,1,w) \equiv P(S=s, R=1, W=w \mid \text{evidencia})\ \forall s,w,$ son todas 0.0 (y no se muestran) ya que, si llueve ($R=2$), la probabilidad conjunta para cualquier situación donde $R=1$ es 0.0 (¡pues se sabe que llueve!). Ante esta situación BNT opta por no devolver (ni mostrar) todos estos ceros. Al igual que se ha visto anteriormente, pueden obtenerse todos los resultados de la siguiente forma:

```
m = marginal_nodes(motor, [S R W], 1)
m =
    domain: [2 3 4]
         T: [2x2x2 double]
    ...
m.T
ans(:,:,1) =
            0       0.0820
            0       0.0018
ans(:,:,2) =
            0       0.7380
            0       0.1782
```

Explicación más probable

Dada una evidencia, la función `calc_mpe` de BNT permite obtener la "explicación más probable" (es decir, el valor más probable de la variable aleatoria de cada nodo, posiblemente dada alguna evidencia). Esta función también obtiene la log-verosimilitud correspondiente. Por ejemplo, si no se especifica ninguna evidencia:

```
evidencia = cell(1,N);
[explMaxProb, logVerosim] = calc_mpe(motor, evidencia)
explMaxProb =
    [2]    [1]    [2]    [2]
logVerosin =
   -1.1270
```

Es decir, $C=2$, $S=1$, $R=2$, $W=2$. Al no haber ningún nodo observado (ninguna evidencia), `calc_mpe` obtiene la *moda* de la distribución conjunta de todas las variables; es decir, una combinación de valores de variables, para la que la probabilidad conjunta es máxima. En este caso, según las tablas de probabilidad especificadas en la Figura 1 la situación conjunta más probable es: que esté nublado, que el aspersor esté parado, que llueva y que el césped esté húmedo.

Es interesante observar que, si se especifica cualquier evidencia parcial que no contradiga la explicación más probable, se seguirá obteniendo el mismo resultado:

```
evidencia{W} = 2;
[explMaxProb, logVer] = calc_mpe(motor, evidencia)
explMaxProb =
    [2]    [1]    [2]    [2]
logVer =
   -1.1270
```

2.3. Aprendizaje

En esta sección vamos a explorar las capacidades de aprendizaje principales que ofrece BNT: aprendizaje con datos completos y aprendizaje con datos incompletos mediante Esperanza Maximización (EM).

Para ello seguiremos con el ejemplo del *"Sprinkler"*. Una vez definida la red y sus tablas de probabilidades, se pueden generar muestras aleatorios de "observaciones"; es decir, de valores de las variables aleatorias de los nodos. Para ello BNT ofrece la función `sample_bnet()`, que toma como argumento una red bayesiana completamente definida y, siguiendo la distribución de probabilidad conjunta definida por la red, produce una muestra aleatoria. Esta muestra contiene, para cada nodo de la red, un valor de su variable aleatoria correspondiente.

Aprendizaje a partir de datos completos

Para empezar, inicializamos el generador de números aleatorios y generamos 100 datos completos a partir de la red `redB`, almacenándolos en el *cell array* `samples`:

```
semilla = 0; rng(semilla);
nMuestras = 100;
muestras = cell(N, nMuestras);
for i=1:nMuestras muestras(:,i) = sample_bnet(redB); end
```

Conviene visualizar estas muestras a ver si nos parecen razonables según las distribuciones definidas para esta red (Fig.1):

```
muestras'
ans =
  1004 cell array
    [2]    [1]    [2]    [2]
    [2]    [1]    [1]    [1]
    [2]    [1]    [2]    [2]
    [2]    [1]    [2]    [2]
    [1]    [2]    [2]    [2]
    ...    ...    ...    ...
```

A partir de estos datos, estimaremos las probabilidades de otra red, con la misma estructura que `redB`. Para ello, primero creamos una red vacía con esta estructura:

```
redAPR        = mk_bnet(grafo, tallaNodos);
redAPR.CPD{C} = tabular_CPD(redAPR, C);
redAPR.CPD{R} = tabular_CPD(redAPR, R);
redAPR.CPD{S} = tabular_CPD(redAPR, S);
redAPR.CPD{W} = tabular_CPD(redAPR, W);
```

Y ahora usamos las observaciones de `muestras` para estimar las distribuciones de cada nodo. En este ejemplo simple, la estimación es inmediata: basta contar el número de veces que un valor de variable aleatoria aparece en la muestra, y normalizar por el número de muestras. La función de BNT `learn_params()` hace esta estimación trivial:

```
redAPR2=learn_params(redAPR, muestras);
```

Se pueden mostrar las probabilidades estimadas como sigue:

```
TPCaux = cell(1,N);
for i=1:N s=struct(redAPR2.CPD{i}); TPCaux{i}=s.CPT; end
```

Podemos ver, por ejemplo, la probabilidad estimada para el nodo `W` (el 4):

```
dispcpt(TPCaux{W})
1 1 : 1.0000 0.0000
2 1 : 0.0556 0.9444
1 2 : 0.0435 0.9565
2 2 : 0.0000 1.0000
```

Ahora se puede comparar esta distribución con la correspondiente de la Fig.1. Si hacemos lo mismo para el resto de distribuciones, observaremos que con las 100 muestras de aprendizaje se han estimado con bastante exactitud las probabilidades de la red original, de la que se extrajeron las muestras aleatoriamente.

Aprendizaje con datos incompletos mediante EM

Si los datos de aprendizaje no contienen valores para todas las variables aleatorias se dice que estos datos son *"incompletos"*. El caso más usual de datos incompletos es cuando en todas las muestras falta el valor de una variable aleatoria concreta (siempre la misma). En este caso se dice que dicha variable es *"latente"* u *"oculta"*. En dichos casos también se pueden estimar las probabilidades de todos los nodos de la red mediante Esperanza Maximización (EM). En BNT el aprendizaje EM está implementado mediante la función `learn_params_em()`. Este modo de aprendizaje también es capaz de estimar todas las probabilidades de los nodos de una red si los variables aleatorias ocultas son diferentes para las sucesivas muestras de aprendizaje. En el ejemplo siguiente se explora esta capacidad de BNT.

Para crear datos incompletos, copiamos `muestras` en `muestrasS` y "ocultamos" aleatoriamente un cierto porcentaje (por ejemplo el 50 %) de los valores de variables aleatorias:

```
muestrasS = muestras;
semilla = 0; rng(semilla);
ocultas   = rand(N, nMuestras) > 0.5;
[I,J]     = find(ocultas);
for k=1:length(I) muestrasS{I(k), J(k)} = []; end
```

Podemos comprobar qué datos se han "ocultado":

```
muestrasS'
ans =
  1004 cell array
     []     []    [2]     []
     []    [1]    [1]     []
     []     []    [2]     []
     []    [1]     []    [2]
    [1]     []     []     []
    ...    ...    ...    ...
```

Preparamos una nueva red cuyas probabilidades van a ser re-estimadas por EM:

```
redEM = mk_bnet(grafo, tallaNodos, 'discrete', nodosDiscretos);
redEM.CPD{C} = tabular_CPD(redEM, C);
redEM.CPD{R} = tabular_CPD(redEM, R);
redEM.CPD{S} = tabular_CPD(redEM, S);
redEM.CPD{W} = tabular_CPD(redEM, W);
motorEM      = jtree_inf_engine(redEM);
```

Y procedemos al aprendizaje por EM:

```
maxIter = 100; eps = 1e-4;
semilla = 0; rng(semilla);
[redEM2, trazaLogVer] = learn_params_em(motorEM, muestrasS, maxIter, eps);
auxTPC = cell(1,N);
for i=1:N s=struct(redEM2.CPD{i}); auxTPC{i}=s.CPT; end
```

Probabilidades estimadas para los nodos `S` y `W` tras 20 iteraciones de EM:

```
dispcpt(auxTPC{S})
1 : 0.6676 0.3324
2 : 0.9839 0.0161

dispcpt(auxTPC{W})
1 1 : 0.9996 0.0004
2 1 : 0.0340 0.9660
1 2 : 0.0071 0.9929
2 2 : 0.1329 0.8671
```

Análogamente podemos visualizar las probabilidades estimadas para los otros nodos, `C,R`, y comparar los resultados obtenidos con las distribuciones de los nodos de la red original (Fig.1).

2.4. Ejercicios propuestos (a entregar)

A) En el ejemplo *"Sprinkler"*, repetir los procesos de aprendizaje a partir de datos completos y de datos incompletos explicados en la Sec.2.3, usando un número mucho mayor de muestras de aprendizaje (por ejemplo, `nMuestras = 1000`), así como permitiendo un mayor número de iteraciones. Comentar los resultados obtenidos.

B) En la página 6.10 del tema 6 de teoría sobre modelos gráficos probabilísticos podemos ver una sencilla red bayesiana para diagnóstico de cáncer de pulmón. Se pide:

1. desarrollar un script `matlab` que implemente dicha red usando `BNT`
2. ¿Cuál es la probabilidad de que un paciente no fumador no tenga cáncer de pulmón si la radiografía ha dado un resultado negativo pero sufre disnea?
3. ¿Cuál es la explicación más probable de que un paciente sufra cáncer de pulmón?