

¿Son enredadas las **Redes Bayesianas**?

Redes Bayesianas a partir de una aplicación en R

Joshua Kunst Fuentes

14 de agosto de 2008

Motivación

Este presentación tiene como objetivo presentar de forma sencilla el porque de una *Red Bayesiana*. Además de presentar su implementación en R con la librería `Dea1`, con el fin de manejar de buena manera las aplicaciones y comandos del paquete.

Introducción

Dea1 es un paquete para usar en R. Incluye muchos métodos para analizar datos usando *Redes Bayesianas* tanto con variables discretas como continuas.

Esquema

- 1 Preliminares
 - ¿Por qué una Red Bayesiana?
 - Grafo Acíclico Dirigido
 - Red Bayesiana
- 2 Aplicación
 - La Base de Datos: Rats
 - Creando la Red
 - Otorgando Estructura a la Red
 - Aprendizaje
 - Buscando el Mejor Modelo
- 3 Un Ejemplo
 - Los Datos
 - El Código
 - El Resultado

¿Por qué una Red Bayesiana?

- 1 Las **Redes Bayesianas** son un modelo probabilístico que representa de forma fácil e intuitiva relaciones de dependencia entre variables aleatorias.
- 2 El objetivo fundamental de aplicar este modelo a un fenómeno real es encontrar la verdadera **dependencia** de las variables que se involucran en el fenómeno de interés.

¿Por qué una Red Bayesiana?

- 1 Las **Redes Bayesianas** son un modelo probabilístico que representa de forma fácil e intuitiva relaciones de dependencia entre variables aleatorias.
- 2 El objetivo fundamental de aplicar este modelo a un fenómeno real es encontrar la verdadera **dependencia** de las variables que se involucran en el fenómeno de interés.

¿Por qué una Red Bayesiana?

- 1 Las **Redes Bayesianas** son un modelo probabilístico que representa de forma fácil e intuitiva relaciones de dependencia entre variables aleatorias.
- 2 El objetivo fundamental de aplicar este modelo a un fenómeno real es encontrar la verdadera **dependencia** de las variables que se involucran en el fenómeno de interés.

Grafo Acíclico Dirigido

Definición

Un **Grafo Acíclico Dirigido** (*GAD*) es un par $(\mathcal{V}, \mathcal{E})$, en donde \mathcal{V} representa un conjunto de vertices y \mathcal{E} es un conjunto de vertices dirigidos, tal que no posee un circuito.

Este grafo acíclico representará la estructura de dependencia de las variables en la Red Bayesiana.

Grafo Acíclico Dirigido

Definición

Un **Grafo Acíclico Dirigido** (*GAD*) es un par $(\mathcal{V}, \mathcal{E})$, en donde \mathcal{V} representa un conjunto de vertices y \mathcal{E} es un conjunto de vertices dirigidos, tal que no posee un circuito.

Este grafo acíclico representará la estructura de dependencia de las variables en la Red Bayesiana.

Red Bayesiana

Definición

*Una **Red Bayesiana** es un grafo acíclico dirigido tal que cada vertice (o nodo) es una variable aleatoria cuya distribución está condicionada por los valores de sus padres.*

Nota: Un nodo j es un padre de un vértice i si existe una flecha dirigida $j \rightarrow i$ en el grafo.

Propiedades

La distribución conjunta de la Red Bayesiana es:

$$\Pr\{X_i = x_i\}_{i=1}^n = \prod_{i=1}^n \Pr\{X_i = x_i | Pa(X_i)\}$$

Red Bayesiana

Definición

*Una **Red Bayesiana** es un grafo acíclico dirigido tal que cada vertice (o nodo) es una variable aleatoria cuya distribución está condicionada por los valores de sus padres.*

Nota: Un nodo j es un padre de un vértice i si existe una flecha dirigida $j \rightarrow i$ en el grafo.

Propiedades

La distribución conjunta de la Red Bayesiana es:

$$\Pr\{X_i = x_i\}_{i=1}^n = \prod_{i=1}^n \Pr\{X_i = x_i | Pa(X_i)\}$$

Red Bayesiana

Definición

Una Red Bayesiana es un grafo acíclico dirigido tal que cada vertice (o nodo) es una variable aleatoria cuya distribución está condicionada por los valores de sus padres.

Nota: Un nodo j es un padre de un vértice i si existe una flecha dirigida $j \rightarrow i$ en el grafo.

Propiedades

La distribución conjunta de la Red Bayesiana es:

$$\Pr\{X_i = x_i\}_{i=1}^n = \prod_{i=1}^n \Pr\{X_i = x_i | Pa(X_i)\}$$

Rats es una base de datos artificial. 24 ratas, 12 machos y 12 hembras, fueron expuestas a una de tres drogas para perder peso. A cada una se le registró dos veces la pérdida de peso. Así, la base de dato consta de dos variables discretas: el sexo y tipo de droga; y dos variables continuas: el primer y segundo registro de peso.

`Dea1` recibe los datos especificados de la forma usual en R, como los son los archivos de texto plano. De este modo cargaremos los datos a utilizar:

```
rats.df <- read.table(rats.dat, header = T)
```

Rats es una base de datos artificial. 24 ratas, 12 machos y 12 hembras, fueron expuestas a una de tres drogas para perder peso. A cada una se le registró dos veces la pérdida de peso. Así, la base de dato consta de dos variables discretas: el sexo y tipo de droga; y dos variables continuas: el primer y segundo registro de peso.

Dea1 recibe los datos especificados de la forma usual en R, como los son los archivos de texto plano. De este modo cargaremos los datos a utilizar:

```
rats.df <- read.table(rats.dat, header = T)
```

En `Dea1`, para trabajar los datos, debe ser inicializada una red con el siguiente comando que recibe como argumento los datos:

```
rats <- network(rats.df)
```

Esto creará un objeto de la clase `Network`, el cual tiene ciertos atributos como los nodos:

```
rats.nd <- nodes(rats)  
rats.nd[[1]]  
rats.nd$Drug
```

En `Dea1`, para trabajar los datos, debe ser inicializada una red con el siguiente comando que recibe como argumento los datos:

```
rats <- network(rats.df)
```

Esto creará un objeto de la clase `Network`, el cual tiene ciertos atributos como los nodos:

```
rats.nd <- nodes(rats)  
rats.nd[[1]]  
rats.nd$Drug
```

La estructura de dependencia se le puede brindar con el comando:

```
rats <- network(rats.df, specifygraph = T,  
inspectprob = T)
```

La opción `specifygraph = T` abrirá una ventana mostrando los vértices en donde, a partir de *clicks* entre nodos, se podrá trazar el grafo que representará la estructura de la red.

La opción `inspectprob = T`, luego de haber graficado la red, abrirá una ventana con la cual se podrá inspeccionar la distribución de probabilidad sugerida por el software. La probabilidad local es modificable con el comando `localprob()`.

La estructura de dependencia se le puede brindar con el comando:

```
rats <- network(rats.df, specifygraph = T,  
inspectprob = T)
```

La opción `specifygraph = T` abrirá una ventana mostrando los vértices en donde, a partir de *clicks* entre nodos, se podrá trazar el grafo que representará la estructura de la red.

La opción `inspectprob = T`, luego de haber graficado la red, abrirá una ventana con la cual se podrá inspeccionar la distribución de probabilidad sugerida por el software. La probabilidad local es modificable con el comando `localprob()`.

La estructura de dependencia se le puede brindar con el comando:

```
rats <- network(rats.df, specifygraph = T,  
inspectprob = T)
```

La opción `specifygraph = T` abrirá una ventana mostrando los vértices en donde, a partir de *clicks* entre nodos, se podrá trazar el grafo que representará la estructura de la red.

La opción `inspectprob = T`, luego de haber graficado la red, abrirá una ventana con la cual se podrá inspeccionar la distribución de probabilidad sugerida por el software. La probabilidad local es modificable con el comando `localprob()`.

Asignando la distribución a **priori**

Es necesario otorgar las probabilidades locales puesto que de este modo se podrá obtener la priori conjunta de la red.

Ya asignadas las probabilidades locales se puede acceder a ellas con el comando `jointprior()` que se utiliza recibiendo como argumento la red.

```
rats.j <- jointprior(rats)
```

Asignando la distribución a **priori**

Es necesario otorgar las probabilidades locales puesto que de este modo se podrá obtener la priori conjunta de la red.

Ya asignadas las probabilidades locales se puede acceder a ellas con el comando `jointprior()` que se utiliza recibiendo como argumento la red.

```
rats.j <- jointprior(rats)
```

Obteniendo la distribución a **posteriori**

Ya establecido el modelo, vale decir, el grafo propuesto, la distribución local y por ende la distribución conjunta; se procede a realizar inferencia: Obtener la distribución a posteriori.

```
rats <- getnetwork( learn(rats, rats.df, rats.j ) )
```

Este nuevo objeto posee las probabilidades a priori, posteriori.

Ejemplo:

```
localprior( nodes(rats)$Sex)  
localposterior( nodes(rats)$Sex)
```

Obteniendo la distribución a **posteriori**

Ya establecido el modelo, vale decir, el grafo propuesto, la distribución local y por ende la distribución conjunta; se procede a realizar inferencia: Obtener la distribución a posteriori.

```
rats <- getnetwork( learn(rats, rats.df, rats.j ) )
```

Este nuevo objeto posee las probabilidades a priori, posteriori.

Ejemplo:

```
localprior( nodes(rats)$Sex)  
localposterior( nodes(rats)$Sex)
```

Una cosa que hay que tener en cuenta es que nuestro modelo propuesto a priori puede que no sea el que mejor se ajuste a los datos observados. Un objetivo fundamental es encontrar la verdadera estructura, o por lo menos, la que mejor se ajuste a los datos obtenidos.

La alternativa a este inconveniente es calcular todos los posibles modelo (todos los GAC!) y ordenarlos según su puntaje, el cual mide que tan bien el GAC representa la independencia condicional. Ejemplo:

```
score(rats)
allrats <- networkfamily( rats.df, rats, rats.j)
allrats <- nwsort( getnetwork(allrats) )
```

Una cosa que hay que tener en cuenta es que nuestro modelo propuesto a priori puede que no sea el que mejor se ajuste a los datos observados. Un objetivo fundamental es encontrar la verdadera estructura, o por lo menos, la que mejor se ajuste a los datos obtenidos.

La alternativa a este inconveniente es calcular todos los posibles modelo (todos los GAC!) y ordenarlos según su puntaje, el cual mide que tan bien el GAC representa la independencia condicional. Ejemplo:

```
score(rats)
allrats <- networkfamily( rats.df, rats, rats.j)
allrats <- nwsort( getnetwork(allrats) )
```

Lo anterior se puede complicar si hay muchas variables en estudio, lo que hace que el método no sea del todo óptimo, pues el número de redes crece exponencialmente al aumentar el número de nodos. Note que con 5 nodos se puede obtener hasta 29.281 redes distintas.

Deal tiene integrado dos métodos que a partir de una red inicial aleatoriamente escoge otra levemente (la perturba para elevar el puntaje de la red actual) distinto calculando su *score* para luego volver a modificar esta última y seguir cierta cantidad de veces, hasta que entrega la red con mayor puntaje:

```
rats.h <- getnetwork( heuristic(rats, rats.df,  
rats.j, restart = 20, degree = 5))  
rats.h <- getnetwork( autosearch(rats, rats.df,  
rats.j, maxiter = 100))
```

Lo anterior se puede complicar si hay muchas variables en estudio, lo que hace que el método no sea del todo óptimo, pues el número de redes crece exponencialmente al aumentar el número de nodos. Note que con 5 nodos se puede obtener hasta 29.281 redes distintas.

Deal tiene integrado dos métodos que a partir de una red inicial aleatoriamente escoge otra levemente (la perturba para elevar el puntaje de la red actual) distinto calculando su *score* para luego volver a modificar esta última y seguir cierta cantidad de veces, hasta que entrega la red con mayor puntaje:

```
rats.h <- getnetwork( heuristic(rats, rats.df,  
rats.j, restart = 20, degree = 5))  
rats.h <- getnetwork( autosearch(rats, rats.df,  
rats.j, maxiter = 100))
```

Para los datos:

Ambiente	Temperatura	Humedad	Viento	Clase
Soleado	Alta	Alta	No	No
Soleado	Alta	Alta	Si	No
Nublado	Alta	Alta	No	Si
Lluvia	Media	Alta	No	Si
Lluvia	Baja	Normal	No	Si
Lluvia	Baja	Normal	Si	No
Nublado	Baja	Normal	Si	Si
Soleado	Media	Alta	No	No
Soleado	Baja	Normal	No	Si
Lluvia	Media	Normal	No	Si
Soleado	Media	Normal	Si	Si
Nublado	Media	Alta	Si	Si
Nublado	Alta	Normal	No	Si
Lluvia	Media	Alta	Si	No

Las siguientes líneas de comando en R:

```
datos <- read.table(clipboard; header = T)
datos
red <- network(datos)
red.j <- jointprior(red)
red <- getnetwork( learn(red, datos, red.j ) )
red.s <- getnetwork( autosearch( red, datos , red.j,
maxiter = 100) )
red.h <- getnetwork( heuristic ( red, datos , red.j,
restart = 5, degree = 10))
plot(red.s)
plot(red.h)
```

