Modelos Gráficos Probabilísticos - 2024-2025

Master Ciencia de Datos

Universidad de Granada

Práctica 3

Miguel García López

Índice

Índice de figuras

Índice de cuadros

1. Introducción

El objetivo de esta práctica es aplicar técnicas de aprendizaje de redes bayesianas utilizando el lenguaje de programación R, haciendo uso de diversos paquetes especializados como `bnlearn`, `gRain`, y `Rgraphviz`.

A lo largo de este trabajo se abordarán aspectos esenciales del aprendizaje de redes bayesianas, tales como la estimación de parámetros mediante distribuciones de probabilidad condicional, los tests de independencia, y el uso de distintos *scores* para la evaluación de modelos. Asimismo, se explorará el proceso de aprendizaje de la estructura del grafo acíclico dirigido (DAG) que subyace a la red, a través de métodos basados en pruebas estadísticas y en medidas de evaluación.

2. Red seleccionada

La red **SACHS** (figura 1) es una red bayesiana derivada de datos experimentales sobre proteínas y señales celulares. Fue presentada por K. Sachs et al. en su estudio sobre inferencia de redes causales en sistemas biológicos. Esta red es ampliamente utilizada como referencia en el aprendizaje de redes bayesianas, debido a su complejidad intermedia y su relevancia en biología de sistemas.

La red contiene un total de 11 nodos, cada uno representando una proteína o una señal biológica, y 17 arcos dirigidos que modelan las relaciones causales o dependencias condicionales entre estas variables. Además, cuenta con un total de 178 parámetros necesarios para definir las distribuciones de probabilidad condicional asociadas a cada nodo. La estructura de la red presenta un *grado medio* y un *tamaño medio del manto de Markov* de aproximadamente 3,09, reflejando una conectividad moderada entre los nodos. El grado máximo de entrada (número máximo de padres de un nodo) es 3, indicando que ningún nodo depende directamente de más de tres variables.

Cuadro 1: Resumen de los parámetros de la red SACHS

Característica	**Valor**
Nombre de la red	SACHS
Número de nodos	11
Número de arcos (relaciones)	17
Número total de parámetros	178
Tamaño medio del manto de Markov	3.09
Grado medio	3.09
Máximo grado de entrada (in-degree)	3

Los archivos de la red están disponibles en varios formatos compatibles con herra-

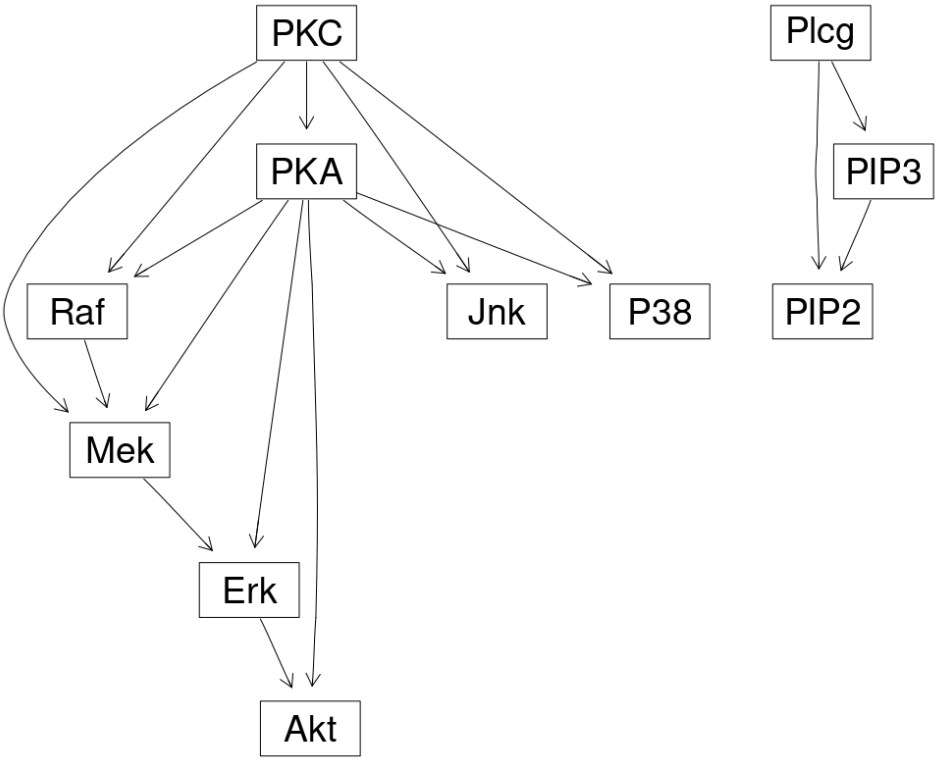

Figura 1: DAG de la red SACHS

mientas de aprendizaje y análisis de redes bayesianas, tales como BIF, DSC, NET y RDA/RDS para objetos `bn.fit` en R.

3. Estructuras obtenidas

En esta sección, se realizan simulaciones de datos, el aprendizaje de la estructura mediante distintos algoritmos y la comparación de los modelos obtenidos respecto a la red original.

Para evaluar el comportamiento de los algoritmos de aprendizaje de estructuras de redes bayesianas, se simulan dos conjuntos de datos de distinto tamaño a partir de la red original SACHS:

- Un primer conjunto de datos con 200 observaciones.
- Un segundo conjunto de datos con 5000 observaciones.

La simulación se realiza mediante la función `rbn()`, que permite generar datos sintéticos a partir de una red bayesiana previamente definida.

```r
# Simular conjunto de datos con 200 casos
data_200 <- rbn(sachs, n = 200)
# Simular conjunto de datos con 5000 casos
data_5000 <- rbn(sachs, n = 5000)

# Mostrar las primeras filas de cada conjunto de datos
head(data_200)
head(data_5000)
```

Se procede a aprender la estructura de la red a partir de los datos simulados, empleando dos métodos distintos:

1. **Método basado en scores**: utiliza algoritmos de búsqueda, como Hill-Climbing (`hc`), evaluando las estructuras mediante una función de puntuación.
2. **Método basado en tests de independencia**: utiliza algoritmos como `iamb`, que se basan en pruebas de independencia condicional para determinar la estructura.

Los algoritmos se aplican a ambos conjuntos de datos (200 y 5000 observaciones) para evaluar cómo afecta el tamaño de la muestra al resultado.

```r
# Metodo basado en scores (Hill-Climbing)
learned_score_200 <- hc(data_200)
modelstring(learned_score_200)
graphviz.plot(learned_score_200)

learned_score_5000 <- hc(data_5000)
modelstring(learned_score_5000)
graphviz.plot(learned_score_5000)

# Metodo basado en tests de independencia (IAMB)
learned_independence_200 <- iamb(data_200)
learned_independence_200 <- cextend(learned_independence_200)
modelstring(learned_independence_200)
graphviz.plot(learned_independence_200)

learned_independence_5000 <- iamb(data_5000)
learned_independence_5000 <- cextend(learned_independence_5000)
modelstring(learned_independence_5000)
graphviz.plot(learned_independence_5000)
```

Se procede a comparar primero las redes obtenidas por medio de un algoritmo de *score*. En las figuras 2 y 3 se pueden observar las dos redes obtenidas tras realizar el ajuste con los datos simulados.

Lo primero que salta a la vista es la división de dos grafos principales, el grafo parece estar dividido por dos sub-grafos desconectados entre si. Esto se ha visto reflejado en ambos DAGs obtenidos. Si bien el grafo más pequeño no refleja la relación completa entre las variables $PIP2, PIP3$ y $Plcg$, ha podido representar la separación con el resto

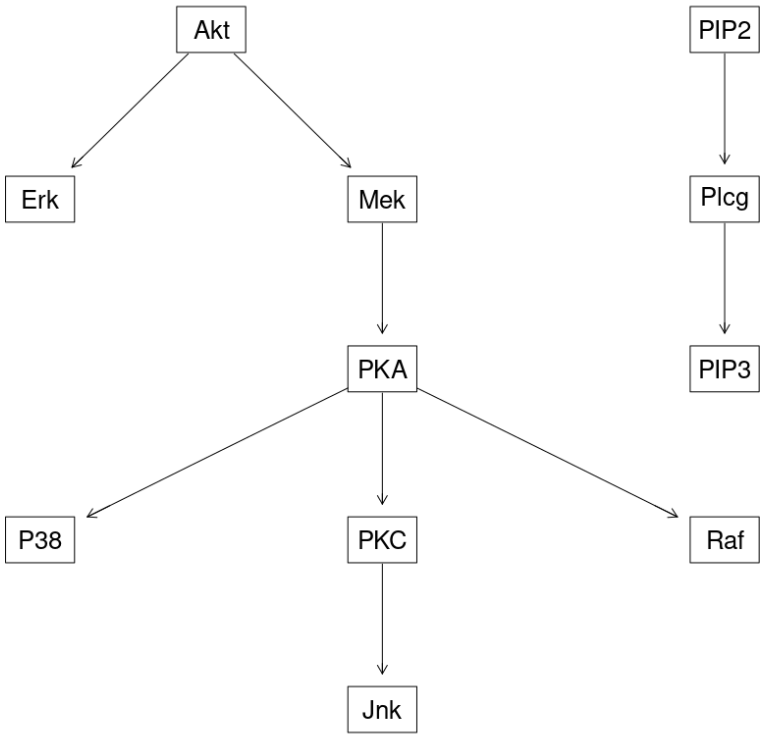

Figura 2: DAG aprendido con método de *score* con 200 ejemplos.

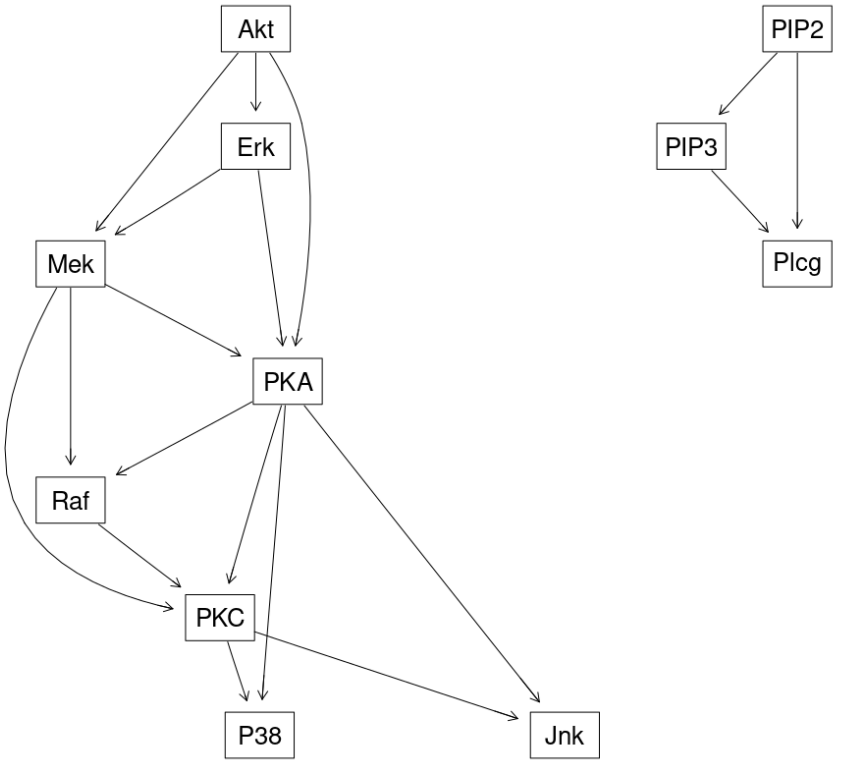

Figura 3: DAG aprendido con método de *score* con 5000 ejemplos.

de variables. En el grafo más grande se obtiene un DAG inverso al original, pues las aristas están conectadas correctamente, pero orientadas en sentido contrario.

Esto es debido a que muchos DAGs equivalentes a nivel de independencia condicional pueden diferir solo en la orientación de sus aristas. Si dos grafos representan las mismas independencias condicionales, los algoritmos de estructura no siempre pueden decidir la dirección correcta solo con datos (sin intervención o conocimiento a priori).

Ocurre lo mismo en el sub-grafo grande de ambos DAGs (sobre todo el DAG de 5000 ejemplos, pues capta mejor la complejidad del original). Se relacionan variables como PKC con las correctas, pero con las aristas mal orientadas en el sentido de la dirección. En general se captura bien relaciones entre variables, aunque hay algún fallo, como la relación entre Mek y Akt del grafo en la figura 3. En el original esta relación no es inmediata, sino que existe un nodo intermedio que es Erk. Esto significa que el algoritmo ha captado una dependencia estadística real (porque Mek y Akt están relacionados), pero no ha sido capaz de reflejar que esa relación está mediada por Erk.

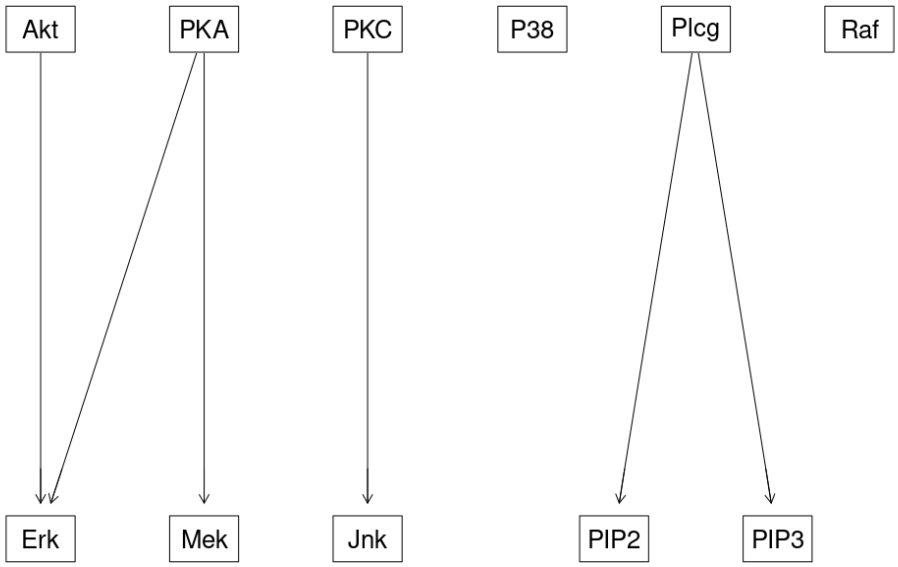

Figura 4: DAG aprendido con método de *tests* de independencia con 200 ejemplos.

En el caso de los grafos aprendidos por medio de un algoritmo que hace uso de *tests* de independencia (figuras 4, 5) puede verse a primera vista que el ajuste en relación con el original es peor. El grafo pequeño no ha sido capaz de captar demasiadas relaciones entre variables, aunque las pocas que ha encontrado están bien orientadas y son correctas. En el grafo grande siguen existiendo problemas de direccionalidad, pero el sub-grafo de tres variables ($PIP2, PIP3, Plcg$) ha sido representado de manera perfecta.

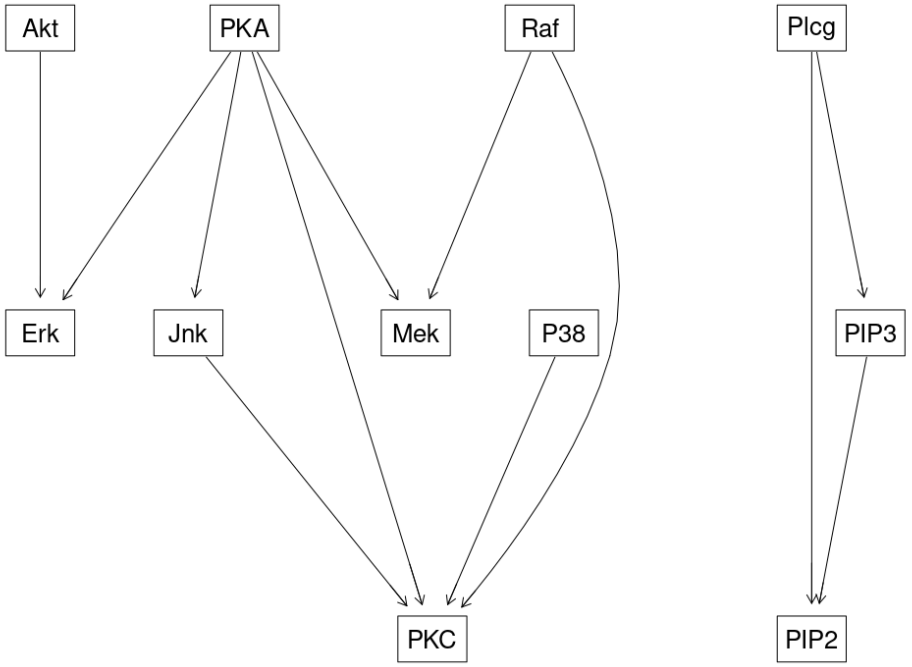

Figura 5: DAG aprendido con método de *tests* de independencia con 5000 ejemplos.

En esta red concreta y dados los pocos experimentos realizados, lo que se puede observar a simple vista es que los grafos obtenidos por medio de algoritmos basados en *score* son mejores aprendiendo la estructura general del DAG. Si bien ambos fallan en la orientación de las relaciones condicionales, el que considero que capta mejor esas relaciones es el `hc`.

4. Parámetros aprendidos

Se procede a aprender los parámetros de las redes por medio del conjunto de datos respectivo a cada red

```
fit_hc_200 <- bn.fit(learned_score_200, data_200)
fit_hc_5000 <- bn.fit(learned_score_5000, data_5000)
fit_iamb_200 <- bn.fit(learned_independence_200, data_200)
fit_iamb_5000 <- bn.fit(learned_independence_5000, data_5000)
```

En la tabla 2, se observa que el modelo original asigna una alta probabilidad a `Erk = LOW` (0.85). Sin embargo, HC con $n = 200$ y con $n = 5000$ estima una probabilidad menor (0.14), indicando pobre ajuste de las probabilidades. Con IAMB, ambos métodos mejoran, acercándose al original. En la Tabla 3, HC con $n = 5000$ recupera mejor la estructura, asignando 0.84 a `AVG`, mientras que IAMB subestima esta probabilidad (0.39).

Modelo	**LOW**	**AVG**	**HIGH**
Original	0.85	0.39	0.01
HC (n=200)	0.14	0.83	0.02
HC (n=5000)	0.14	0.81	0.05
IAMB (n=200)	0.75	0.25	0.00
IAMB (n=5000)	0.94	0.06	0.00

Cuadro 2: Parámetros de `Erk` (Mek = LOW, PKA = LOW).

Modelo	**LOW**	**AVG**	**HIGH**
Original	0.05	0.73	0.22
HC (n=5000)	0.16	0.84	0.00
IAMB (n=5000)	0.05	0.39	0.55

Cuadro 3: Parámetros de `Erk` (Mek = AVG, PKA = AVG).

Modelo	**LOW**	**AVG**	**HIGH**
Original	0.06	0.92	0.02
HC (n=200)	0.05	0.95	0.00
HC (n=5000)	0.37	0.63	0.00

Cuadro 4: Parámetros de `PKA` (PKC = AVG).