Ejemplo de utilización de Markov Logic Network para resolver un problema de IoT

J.I. Estévez-Damas[1]

[a] *Universidad de La Laguna, Spain*

Keywords: Internet Of Things, Representación, Lógica de primer orden, Red lógica de Markov, Ideas preliminares

1. Introducción

Uno de los problemas típicos de la IOT consiste en lograr que los "objetos inteligentes" de la red tomen decisiones a partir de la información obtenida de diversas fuentes: conocimiento previo, evidencias cosechadas por los objetos con capacidad de sensar e interacción con el usuario. Voy a describir y comentar en esta sección un enfoque basado en herramientas propias de la IA:

- **Representación.** La lógica de primer orden será el lenguaje en el que expresaremos el conocimiento previo y las reglas de decisión. Es un lenguaje que permite reflejar de forma precisa información general sobre conjuntos de objetos, y posee esquemas de resolución a partir de los cuáles se pueden realizar inferencias.

- **Manejo de la incertidumbre y las inconsistencias.** En la IoT no podemos esperar un conocimiento preciso y plenamente consistente del mundo. Por ello es necesario emplear herramientas de la IA que faciliten el razonamiento ante esta problemática. En este ensayo proponemos el uso de la denominada "Markov Logic Network" (MLN). Esta forma de representar conocimiento integra la lógica de primer orden con la representación de la incertidumbre, usando la inferencia estadística como fundamento para la inferencia.

- **Aprendizaje.** Nos planteamos analizar un tipo de aprendizaje muy relacionado con el concepto de MLN. Se trata de aprender la "compatibilidad" de reglas predefinidas con las evidencias capturadas. Así,

aquellas reglas más coherentes con la evidencia jugarán un papel mayor en la inferencia y por tanto en la toma de decisiones.

La arquitectura sobre la que vamos a reflexionar inicialmente se compone de un bloque "Sensado del Entorno", un bloque "Base de conocimiento", un bloque "Inferencia" y un bloque "Actuación". Entre el bloque de "Sensado del Entorno" y la "Base de Conocimiento" tenemos el "Interfaz de Sensado" que transforma las información de los sensores en predicados. Este interfaz, también puede utilizar información de la "Base de conocimiento" existente. Entre el bloque "Actuación" y el bloque de "Inferencia" tenemos el "Interfaz de inferencia" que puede ordenar consultas al bloque de inferencia y transformar predicados en acciones a aplicar por el bloque "Actuación".

El modelo de funcionamiento sigue un ciclo simple: el bloque de sensado obtiene información del entorno. El interfaz de sensado realiza a continuación su labor, dando como resultado la actualización de la base de conocimiento. A continuación entra en juego el interfaz de inferencias. Este interfaz realiza un conjunto de consultas al bloque de inferencia. Estas consultas tienen como objetivo obtener información sobre los predicados del tipo "generar". Los predicados de tipo "generar" relacionan tipos de eventos y elementos actuadores, así que básicamente son instancias concretas de acciones. Estas acciones son comunicadas al bloque de "Actuación" que las ejecuta. Esto finaliza el ciclo.

De momento en esta arquitectura no se incluyen bloques de aprendizaje automático, ya que en esta primera parte nos vamos a centrar en la representación y la inferencia. Posteriormente, tras describir la parte correspondiente a la inferencia, nos ocuparemos de las posibilidades de aprendizaje automático.

2. Utilizando la lógica de primer orden

Planteemos el escenario. Tenemos una casa, donde se usan etiquetas electrónicas para generar eventos, hay sensores del ambiente y actuadores, así como elementos con capacidad de procesamiento capaces de acumular la información pertinente para la toma de decisiones (evidencias sobre el estado físico del entorno, base de conocimiento previo). Vamos a crear un modelo de descripción, basado en lógica de primer orden con tipos.

Consideremos los siguientes tipos:

$$\mathcal{T} = \{lugar, sensor, actuador, evento, magnitud, valor, clase\}$$

El conjunto de constantes $\mathcal{C}$ puede ser particionado utilizando los tipos en $\mathcal{T}$.

Además tendremos un conjunto de predicados:

- **localización(sensor/actuador,lugar).** Establece una relación espacial entre un sensor o actuador y un lugar.
- **eventoactivo(evento,sensor).** Señala un evento detectado en el sistema que debe ser procesado
- **despuésde(evento,evento).** Establece una relación temporal entre dos eventos.
- **generar(evento,actuador).** Esta relación implica la producción inmediata de un evento por parte de un actuador.
- **medida(sensor,magnitud,valor).** Establece la medición de un valor para una magnitud por parte de un sensor.
- **tiposensor(sensor,clase).** Permite asignar una o varias clases a un objeto.
- **admiteactuacion(actuador,clase).** Permite asignar una o varias clases a un actuador.

Supongamos ahora que queremos diseñar una regla que permita encender la iluminación de una habitación cuando se produzca la activación del interruptor basado en etiqueta RFID de la habitación y no exista suficiente claridad. Podríamos tener algo así:

$$\begin{aligned} &localizacion(S, H) \wedge tiposensor(S, rfidbasedswitch) \wedge \\ &eventoactivo(defaultevent, S) \wedge localizacion(S1, H) \wedge \\ &medida(S1, lightmeasurement, lowvalue) \wedge localizacion(A, H) \wedge \\ &admiteactuacion(A, roomlightsactuator) \\ &\Rightarrow generar(ev\_turnonroomlights, A) \end{aligned}$$

Bajo la interpretación clásica de la lógica de primer orden, esta regla viene a indicar que cualquier "mundo" donde los predicados grounded del

antecedente de la regla sean verdaderos y el consecuente falso es imposible. Por ejemplo, no puede suceceder:

$$
\begin{aligned}
&localizacion(interruptor13, cocina) = T \\
&tiposensor(interruptor13, rfidbasedswitch) = T \\
&eventoactivo(defaultevent, interruptor13) = T \\
&localizacion(senselight1, cocina) = T \\
&medida(senselightcocina, lightmeasurement, lowvalue) = T \\
&localizacion(smartbulb1, cocina) = T \\
&admiteactuacion(smartbulb, roomlightsactuator) = T
\end{aligned}
$$

y al mismo tiempo:

$$generar(ev\_turnonroomlights, smartbulb) = F$$

Si los predicados del antecedente efectivamente se prueba que son verdaderos (bien, porque sean evidencias o porque se han deducido), entonces, en todos los "mundos" posibles debe ocurrir que:

$$generar(ev\_turnonroomlights, smartbulb) = T$$

Un método de inferencia válido (soundness) y completo (completness) debe ser capaz de obtener esta conclusión.

Un aspecto a considerar se refiere al papel de las constantes en esta clase de representaciones. El análisis de una regla, teniendo en cuenta los objetivos que queremos cumplir puede llevarnos observar que la regla:

- **Puede dar lugar a comportamiento indeseados**. La generalización de la regla podría ser excesiva. Por ejemplo, supongamos que en una habitación hay dos interruptores de tipo $rfidbasedswitch$, uno destinado a conectar la climatización y otro destinado a encender las luces. Según la regla descrita, si el usuario lanza el evento usando el rfid para climatización se inferiría el predicado asociado a la generación del evento "encender luces". Este tipo de problemas, se puede resolver creando tipologías más precisas que describan roles particulares. Introducir esta información en el sistema puede ser cosotoso y no estar exenta de errores, especialmente ante reconfiguraciones del sistema. En el ejemplo que estamos describiendo, el diseñador del sistema asume que todo interruptor rfid está destinado a encender luces en las

habitaciones. El problema se resuelve incluyendo un nuevo tipo "más específico":

$$rfidswitch\_turnonlight$$

A la hora de facilitar el diseño de los roles, lo conveniente sería crear una jerarquía de roles de modo que si por ejemplo, el usuario introduce en la base de conocimiento:

$$tiposensor(interruptor13, rfidswitch\_turnonlight)$$

automáticamente se incluyese la evidencia:

$$tiposensor(interrutpor13, rfidbasedswitch)$$

Esto se puede conseguir mediante inferencia lógica si se incorpora a la base de conocimiento una regla como:

$$tiposensor(X, rfidswitch\_turnonlight) \Rightarrow tiposensor(X, rfidbasedswitch)$$

- **Puede no ser suficientemente descriptiva**. Supongamos ahora que existe una habitación donde hay varios actuadores capaces de encender una luz. Según la regla que hemos escrito, todos estos actuadores producirían su evento asociado. Supongamos que sólo queremos que un tipo concreto de actuador sea el que realice la operación. En este caso se podrían crear tipos de actuación más particulares, por ejemplo podríamos tener en los antecedentes:

$$admiteactuacion(A, roomlightsceiling)$$

Iguamente se puden incluir predicados para categorizar las habitaciones.

Veamos ahora algunas características de la representación en lógica de primer orden con el ejemplo presentado:

- **Se trata de una representación compacta**. Esto se debe al uso de las variables lógicas. Aunque no se representa explícitamente, la fórmula anterior está encabezada por $\forall S, H, S1, A,$ por lo que sería aplicable a todas las habitaciones, sensores y actuadores que cumplan la relaciónes cuya estructura se define en la regla: todo sensor rfid que compata habitación con un actuador catalogado como capaz de dar luz en una habitación y tenga un evento activo va a permitir inferir la generación de un evento de encendido en el actuador. Si este tipo de regla fuera representada usando lógica proposicional habría que haber creado reglas específicas para cada caso, o bien introducir nuevos predicados.

- **Consistencia**. La base de reglas debe ser consistente. Es decir, si hay reglas contradictorias no encontraremos ningún "mundo" posible y no deducirá ningún predicado. Esto complica el diseño de las reglas, ya que en términos "humanos" las especificaciones pueden incluir esta clase de inconsistencias. Esto puede ser ventajosos en algunos ámbitos, pero no en otros. Por ejemplo, supongamos la siguiente regla:

  $$tiposensor(S, smartclock) \wedge eventoactivo(nightdetected, S) \wedge localizacion(S1, H) \wedge tipohabitacion(H, daytimerooms) \wedge tiposensor(S1, movementsensor) \wedge medida(S1, vibration, zerovalue) \Rightarrow generar(ev\_turnoffroomlights, A)$$

  Y además tenemos una regla que impide generar eventos mútuamente excluyentes como:

  $$generar(ev\_turnonlights, A) \Leftrightarrow \neg generar(ev\_turnoffflights, A)$$

  Como vemos, en ese caso podríamos llegar a una contradicción. Evitar estas contradicciones es posible incluyendo más reglas y predicados, pero el precio a pagar es el aumento en la complejidad de la base de reglas.

- **Inclusión en un agente**. Una cuestión a resolver es cómo exactamente incluir esta representación en un agente. Podemos distinguir entre dos aproximaciones. Por una parte, tenemos estrategias en las que el agente mantiene una imagen del estado a partir de dos fuentes:

externamente el conjunto de evidencias se actualiza e internamente infiere hechos sobre el estado a partir de las evidencias y de su base de reglas (al menos en una parte). Por otra parte, tenemos aquellas en las que el agente no infiere los cambios en el estado y se limita a actualizar su base de evidencias desde una fuente externa tras el accionamiento de un actuador.

A la primera aproximación corresponden formalismos como el cálculo situacional o el cálculo de fluentes, esquemas ampliamente usados en robótica, ambos englobados en lo que se conoce métodos de razonamiento sobre acciones. El cálculo situacional tiene como principales ventajas, la inferencia de características del mundo sin necesidad de fuentes externas que lo establezcan. Esto le permite incorporar más información con diferentes grados de abstracción, lo que permite razonamientos más complejos. Por ejemplo, en el mundo de los bloques podríamos tener un sensor para determinar exactamente la posición de cada bloque. El sistema podría inferir internamente la altura de las torres de bloques y cómo esa altura cambia al reposicionar los bloques. El concepto de altura de la torre, es una abstracción que permite definir mecanismos de decisión más compactos para determinados problemas. Otra ventaja es la posibilidad de detectar fallos en el funcionamiento del agente. Supongamos la regla:

$$
\begin{aligned}
&ocalizacion(S, H) \wedge tiposensor(S, lightsensor) \wedge \\
&localizacion(A, H) \wedge admiteactuacion(A, roomlightsactuator) \wedge \\
&turnedon(A) \wedge medida(S, lightmeasurement, lowvalue) \\
&\Rightarrow faultdetection(A, S, H)
\end{aligned}
$$

Esta regla va a determinar que en cualquier habitación donde haya un actuador del tipo *roomlightsactuator*, que se ha activado, si hay un sensor de luz detectando un valor bajo, se infiere una detección de un posible fallo. La clave para lograr esta detección es la actualización del estado después de haber activado el actuador, es decir, debe aparecer el predicado turnedon con el actuador correspondiente en la base de conocimiento. El cálculo situacional establece un marco para razonar con acciones que alteran el estado, permitiendo incluir y eliminar predicados de la base de conocimiento.

En este documento, sin embargo, nos vamos a centrar en la segunda de las aproximaciones. Vamos a suponer, que el agente sólo modifica su

base de conocimiento mediante una fuente externa. Esto simplificará el conjunto de reglas permitièndonos centrarnos en las posibilidades del resto de tecnologías que queremos explorar. En la arquitectura descrita en la introducción, se correspondería con el "Interfaz de sensado".

3. Representación mediante una Markov Logic Network

La base de reglas puede verse como la definición de un conjunto de restricciones que limitan los "mundos" posibles (asignaciones de verdad a los predicados "grounded", donde las variables son sustituidas por constantes). Así, hay asignaciones para los que el conjunto de fórmulas que componen la base de conocimiento son verdaderas y otras en las que no. Las asignaciones que mantienen el valor "True" en todas las fórmulas de la base de conocimiento constituyen los "mundos" posibles.

Las redes lógicas de Markov o MLN introducen otra filosofía diferente. Una MLN también contiene un conjunto de fórmulas en lógica de primer orden, y también establece un grado de verdad para los diferentes mundos. La diferencia es que el grado de verdad asignado no es "T" o "F", sino un valor de **probabilidad**. Así, simplemente habrá mundos más probables que otros. A la hora de consultar si la base de conocimiento puede derivar una fórmula, lo que se obtiene es una probabilidad para la fórmula. En realidad, la definición de la MLN es el conjunto de fórmulas con un peso asignado a cada uno de ellas. El significado de este peso, es que mientras mayor sea, la fórmula asociada tendrá que cumplirse para aquellos mundos que tengan una mínima probabilidad. En el caso límite, si todos los pesos tienden a infinito volvemos a la lógica clásica de primer orden.

La asignación de probabilidad a los mundos, es decir, la definición de una distribución de probabilidad para la asignación de verdad al mundo, se basa en un modelo matemático usado en otros campos (por ejemplo, física), denominado red de Markov (MN). La red de Markov es un grafo no dirigido formado por nodos y arcos que únen estos nodos. Se trata de un modelo gráfico de probabilidad, junto con un conjunto de pesos define una distribución de probabilidad del estado conjunto de los nodos de la red. Esta distribución está factorizada y cada factor considera sólo el estado de los nodos incluidos en uno de los cliqués de la MN (recordemos que un cliqué en un grafo es un subconjunto de los nodos, tales que existe un arco entre cualquier par).

Así, una red *l*ógica de Markov, con su conjunto de fórmulas pesadas y las constantes que pueden sustituir a las variables lógicas, puede verse

como un generador de una red de Markov donde cada nodo de esta última es un posible predicado grounded. Los cliqués se forman a partir de las fórmulas, ya que cada fórmula genera un conjunto de fórmulas grounded y cada una de éstas establece la interconexión por arcos de todos los pares de predicados grounded que la forman. Sin profundizar más, ya podemos intuir que el tamaño de la red de Markov subyacente crece con el número de predicados grounded haciendo más costoso computacionalmente cualquier cálculo de inferencia probabilística.

Veamos algunas características de esta representación:

- **Las fórmulas que se incluyen en la base de conocimiento, no tienen por qué ser siempre ciertas.** El ejemplo estándar que se suele poner es el siguiente:

$$0.8\ friends(X, Y) \wedge smokes(X) \Rightarrow smokes(Y)$$
$$0.3\ smokes(X) \Rightarrow cancer(X)$$

  La primera regla nos dice que para cualquier par de amigos, si uno fuma el otro también. El peso asociado es 0.8. Si tenemos en nuestra base de evidencias, los predicados $friends(Bob, Ann)$ y $smokes(Bob)$, al consultar $smokes(Ann)$ lo que se obtiene es una probabilidad, que será más alta o más baja en función de la importancia relativa del peso asociada a la regla, y del resto de reglas incluídas. La segunda regla nos dice que todo el que fuma tiene cancer, pero luego le asígna un peso más bajo. Así, aunque quizás $smokes(Bob)$, tenga una probabilidad relativamente alta, la inferencia $cancer(Bob)$ nos dé una probabilidad que puede ser relativamente baja.

  En este ejemplo, vemos que es posible representar información similar a las "creencias" que no son siempre ciertas pero que nuestra experiencia les otorga cierto grado de verosimilitud.

- **La base de reglas puede incluir fórmulas contradictorias entre si.** Será la importancia relativa de los pesos la que determine cuál de las fórmulas será más relevante. Esto elimina, la necesidad de un modelado "detallado" y cuidadoso requerido en las representaciones basadas en lógica de primer orden para evitar contradicciones.

- **Procedimientos de inferencia.** Hay que tener en cuenta que los procedimientos de inferencia, requieren búsquedas sobre el espacio de los

mundos. Este espacio de búsqueda puede ser enorme, ya que aumenta exponencialmente con el número de posibles predicados "grounded". Sin embargo, existen diferentes procedimientos de inferencia que pueden trabajar con base de reglas de tamaño medio de forma eficiente:

- Es razonable por ejemplo obtener el "mundo" más probable atendiendo a la información introducida por la MLN (o dicho de otra manera, la asignación de los diferentes predicados "grounded" bajo la MLN definida). Se trata de una estimación denominada MAP (maximum a posteriori en términos de estadística bayesiana). Este procedimiento podría servir, por ejemplo, para obtener si un predicado "grounded" es verdadero o falso en el "mundo" más probable. En el modelo que hemos venido utilizando en este documento, bastaría hacer una consulta sobre todos los predicados "grounded" derivados de $generar(E, A)$. Estos predicados serían el conjunto de acciones recomendadas del sistema. Dicho sistema servirá para determinar cuál/es serían las siguientes acciones a aplicar por el módulo de actuación.
- También es posible obtener de forma aproximada probabilidades marginales para fórmulas, incluso probabilidades con condicionales. Para ello se usan técnicas de muestreo (métodos de Montecarlo). Así, para obtener la probabilidad de una fórmula se realiza una estimación sobre una muestra de mundos de la fracción de mundos donde dicha fórmula se cumple respecto al total. El muestreo lógicamente debe realizarse sobre la distribución de probabilidad de los mundos, por lo que los mundos más probables aparecerán en la muestra con mayor "facilidad".

4. Aprendizaje automático

Hasta ahora hemos esbozado un modelo que puede decidir la aplicación de determinadas acciones tomando como base la información sensada del entorno y una red lógica de Markov. Sin embargo, existen dos aspectos que no hemos tratado:

- ¿Cómo se determinan los valores de los pesos asociados a las fórmulas de una MLN?.
- ¿Qúe fórmulas hay que incluir en la MLN?

La utilización de conocimiento previo y un ajuste basado en prueba - error podría ser una posible respuesta a las preguntas anteriores. Una alternativa a esto es el denominado aprendizaje automático. El aprendizaje automático en su modalidad supervisada requiere de:

1. Un modelo a ajustar.
2. Un conjunto de entrenamiento.
3. Un algoritmo de aprendizaje del modelo, dado el conjunto de entrenamiento.

Las preguntas anteriores se refieren justamente a aspectos del modelo. Queremos calcular los pesos o incluir determinadas fórmulas y dejar fuera otras para lograr que lo que del modelo se derive sea lo más cercano posible a lo obtenido en el conjunto de entrenamiento.

La siguiente cuestión sería: ¿quién es nuestro conjunto de entrenamiento?. Para pensar sobre esto, hemos de recordar que nuestra MLN trata de representar la distribución de probabilidad de todos los mundos posibles: habrá mundos más probables que otros y esto está determinado por la MLN.