Aprendizaje de Independencias Específicas del Contexto en Markov Random Fields

Alejandro Edera y Facundo Bromberg

Facultad Regional Mendoza,
Universidad Tecnológica Nacional
{aedera,fbromberg}@frm.utn.edu.ar

Resumen Los modelos no dirigidos o Markov random fields son ampliamente utilizados para problemas que aprenden una distribución desconocida desde un conjunto de datos. Esto es porque permiten representar una distribución eficientemente al hacer explícitas las independencias condicionales que pueden existir entre sus variables. Además de estas independencias es posible representar otras, las Independencias Específicas del Contexto (CSIs) que a diferencia de las anteriores sólo son válidas bajo ciertos valores que pueden tomar subconjuntos de sus variables. Debido a esto son complicadas de representar y aprenderlas desde datos. En este trabajo presentamos un enfoque para representar CSIs en modelos no dirigidos y un algoritmo que las aprende desde datos utilizando tests estadísticos. Mostramos resultados donde los modelos aprendidos por nuestro algoritmo resultan ser mejores o comparables a modelos aprendidos por otros sin utilizar CSIs.

Keywords: Markov random fields, Context-Specific Independence, Ising Model

1. Introducción

Los *Modelos Probabilísticos Gráficos*, o simplemente modelos gráficos, permiten representar una distribución de probabilidad conjunta $p(X)$ sobre el conjunto de variables aleatorias X, por medio de otra distribución equivalente $p_{\Phi}(X)$ factorizada en un conjunto de factores Φ [7], determinados por las independencias condicionales que existen en la distribución p. La ventaja de esta factorización radica en el hecho de que cada factor $\phi \in \Phi$ es una función *potencial* definida sobre un subconjunto de variables, llamado el *alcance* del potencial, resultando en una reducción en la complejidad espacial de almacenar una distribución en memoria. Muchos

problemas en la práctica consisten en aprender una distribución aproximada desde un conjunto de datos muestreados desde una distribución desconocida. Dada las ventajas computacionales asociadas a utilizar modelos gráficos, estos problemas los utilizan durante el proceso de aprendizaje para representar la distribución. En la literatura pueden encontrarse varios enfoques para realizar este aprendizaje desde datos, uno es el enfoque *Independence-Based*. Los algoritmos que implementan este enfoque aprenden un modelo primero aprendiendo sus independencias ejecutando una sucesión de tests estadísticos y en base a estas definen un conjunto de potenciales cuyos parámetros son aprendidos utilizando algún procedimiento de optimización numérica. En este trabajo nos focalizamos en el problema de aprender distribuciones desde datos utilizando modelos no dirigidos con Independencias Específicas del Contexto – *Context-Specific Independencies* (CSIs) [3,9]. Estas, a diferencia de las independencias no contextualizadas, tienen la particularidad de ser válidas dada una determinada asignación a un subconjunto de variables de su condicionante, denominado *contexto* de la independencia. Estas son bastantes intuitivas en problemas que requieren modelos dirigidos debido a la naturaleza de sus dependencias, por esto su gran uso en estos escenarios [4,5,8,10]. Sin embargo, aún no han sido ampliamente adoptadas en escenarios donde se necesitan modelos no dirigidos, por la dificultad de representar las independencias contextuales debido a su naturaleza espacial. De este modo, presentaremos un enfoque para representar independencias contextuales que hace uso de una representación más expresiva de un modelo no dirigido. Luego presentaremos un algoritmo que aprende independencias contextuales con el enfoque Independence-Based. Para evaluar los modelos aprendidos por nuestro algoritmo, tuvimos en cuenta datos generados desde distribuciones representadas por Ising models. Nuestros resultados muestran que los modelos con independencias contextuales resultan ser mejores o comparables con respecto a los modelos sin ellas.

2. Modelos gráficos no dirigidos

Un modelo gráfico para una distribución $p(X)$ consiste de un grafo G y un conjunto de parámetros numéricos $\boldsymbol{\theta}$, donde $G = (V, E)$ es un conjunto de $n = |V|$ nodos enlazados por aristas $(s, t) \in E$ y cada va-

riable aleatoria $X_s \in X$ se asocia con un nodo $s \in V$. Denotamos por x a una asignación a las variables X. Un *clique* es un subgrafo de G donde todos sus nodos están completamente conectados por aristas. El grafo G se denomina la *estructura de independencias* o *estructura* del modelo y codifica eficientemente un conjunto de independencias condicionales entre las variables X. Denotaremos por $X_A \subseteq X$ al conjunto de variables aleatorias asociadas con un conjunto de nodos $A \subseteq V$ y su dominio con $\mathrm{Val}(X_A) = \otimes_{s \in A} \mathrm{Val}(X_s)$ el producto cartesiano de los dominios $\mathrm{Val}(X_s) = \{x_s^1, x_s^2, \ldots, x_s^k\}$. Decimos que las variables X_A son independientes de X_B dado X_C, denotado por $(X_A \perp X_B \mid X_C)$, si y sólo si en G no existe ningún camino desde los nodos A a los nodos B al remover los nodos C. Una distribución donde sus independencias pueden ser representadas de esta manera, se denominan *graph-isomorph*. En este trabajo asumimos este tipo de distribuciones.

Para facilitar nuestra posterior presentación utilizaremos una *representación exponencial* de un modelo no dirigido. Para algún conjunto de indices $\mathcal{I}$, sea $\Phi = \{\phi_\alpha, \alpha \in \mathcal{I}\}$ un conjunto de funciones potencial cuyos alcances se corresponden con cliques de G, entonces definimos un modelo no dirigido como:

$$p_\Phi(X) = \exp\{\sum_{\alpha \in \mathcal{I}} \theta_\alpha \phi_\alpha(X) - Z\}, \tag{1}$$

donde la constante Z sirve para obtener una distribución normalizada. Cada potencial ϕ_α es un mapeo de la forma: $X_{\phi_\alpha} \mapsto \mathbb{R}_+$. En en este trabajo sólo tendremos en cuenta potenciales que son *funciones indicador*, las cuales toman el valor 1 si la asignación x es igual a la α-ésima asignación de las variables de su alcance, denotado por $\mathrm{Scope}[\phi_\alpha]$, en cualquier otro caso toma el valor 0. Dadas estas funciones indicador, entonces definimos el conjunto $\mathcal{I}$ para indexar el conjunto de todas las asignaciones posibles para los cliques $\mathcal{C}$, es decir, $\mathcal{I} = \{1, \ldots, |\xi|\}$, donde ξ es la unión de todas las asignaciones de la forma $\xi = \bigcup_i^{|\mathcal{C}|} \mathrm{Val}(X_{\mathcal{C}_i})$.

Finalmente, es interesante analizar (1) y remarcar que un modelo no dirigido está determinado por su conjunto Φ de funciones indicador. Modificar alguna de ellas, por ejemplo removiendo variables de sus alcances, cambiaría el valor que esta retorna, resultando en un modelo totalmente

distinto. Nuestro enfoque para representar independencias contextuales en modelos no dirigidos consiste en modificar las asignaciones ξ, con lo cual se modifican los alcances de sus funciones indicador asociadas. A continuación daremos detalle de cuáles funciones indicador modificaremos y cómo las modificaremos.

3. Contextualización de un modelo no dirigido

Comencemos extendiendo la definición de independencia condicional agregando contextos. Dado un conjunto de variables X_A y X_B decimos que son *contextualmente independientes* dado X_C y un contexto x_U, denotado por $(X_A \perp X_B \mid X_C, x_U)$, si en G no existe ningún camino desde los nodos A a los nodos B cuando eliminamos los nodos $C \cup U$. Factorizando la distribución $p(X_A, X_B \mid X_C, x_U)$ a $p(X_A \mid X_C, x_U)\ p(X_B \mid X_C, x_U)$ [3]. Nos referiremos como *modelo contextualizado* a un modelo con estas independencias.

Decimos que una asignación x_C es compatible con otra asignación ${x'}_U$ si $x_C = {x'}_{U \cap C}$. De igual manera, sea ξ un conjunto de asignaciones denotamos por $\xi[{x'}_U]$ al subconjunto de asignaciones compatibles con la asignación ${x'}_U$. Definimos al conjunto C de una asignación x_C como su *alcance* y lo denotamos por Scope$[x_C]$.

Entonces dada las asignaciones ξ de los potenciales de un modelo representamos una independencias contextuales según la siguiente propiedad:

Proposición 1 *Dado los conjuntos A, B, C, U y sea ξ un conjunto de asignaciones compatibles con x_U, si se cumple la independencia contextual $(X_A \perp X_B \mid X_{Scope[\xi] \setminus \{A,B,U\}}, x_U)$, entonces las asignaciones ξ las podemos representar como los conjuntos de asignaciones ξ' y ξ'', donde toda asignación $x' \in \xi'$ cumple $Scope[x'] \cap B = \emptyset$ y toda asignación $x'' \in \xi''$ cumple $Scope[x''] \cap A = \emptyset$.*

Demostración. Dado que una distribución condicionada es proporcional a su conjunta y suponiendo que $(X_A \perp X_B \mid X_{\text{Scope}[\xi] \setminus \{A,B,U\}}, x_U)$ entonces por [6] obtenemos que $p(X_A, X_B \mid X_{\text{Scope}[\xi] \setminus \{A,B,U\}}, x_U)$ factoriza en el producto de factores $p(X_A \mid X_{\text{Scope}[\xi] \setminus \{A,B,U\}}, x_U)$ y $p(X_B \mid$

$X_{\text{Scope}[\xi]\setminus\{A,B,U\}}, x_U$). Estos factores pueden ser representados utilizando la representación exponencial (1), con lo cual obtenemos ξ' y ξ''.

□

Entonces teniendo $(X_A \perp X_B \mid X_C, x'_U)$ aplicamos la prop. 1 sobre las asignaciones ξ de algún clique de $\mathcal{C}$ utilizando la operación $\text{Split}(\xi, A, B)$. Esta operación factoriza ξ de acuerdo a la independencia retornando el conjunto $\xi' \cup \xi''$, donde $\xi' = \{x_{\text{Scope}[x]\setminus A} \mid x \in \xi\}$ y $\xi'' = \{x_{\text{Scope}[x]\setminus B} \mid x \in \xi\}$.

4. Context Separation Algorithm

En el alg. 1 mostramos nuestro algoritmo denominado Context Separation (CS), el cual trabaja sobre las asignaciones ξ de las funciones indicador de un modelo representado como (1). El núcleo del algoritmo consiste en proponer una serie de independencias contextuales de la forma $(X_s \perp X_t \mid x_U)$ que son verificadas ejecutando tests estadísticos sobre los datos, que en caso de satisfacerse, son representadas sobre ξ siguiendo la prop. 1. Proponer las independencias implica determinar qué variables X_U del modelo se contextualizarán. La forma más sencilla para determinarlas sería por cada par de variables X_s y X_t seleccionar como X_U a las $n-2$ variables restantes y luego por cada conjunto W de los $2^{|U|}$ conjuntos para U definir sus $|\text{Val}(X_W)|$ contextos. Claramente esta forma se torna intratable a medida que crece n. Nuestro algoritmo propone una aproximación que consiste en tomar cada clique de G y para cada par X_s y X_t de variables, contextualizar solamente con las restantes variables del clique. Una vez finalizadas todas estas verificaciones, el algoritmo retorna un conjunto de asignaciones que representan la *estructura contextualizada* de G.

Hablando a grandes rasgos, CS está formado por dos bucles anidados. El bucle externo, representado por la función `CS`, comienza iterando en la línea 3 por un conjunto de cliques $\mathcal{C}$ bajo algún ordenamiento dado. Por cada clique $\mathcal{C}_i$, la función `build-assigments`, lo representa como el conjunto de asignaciones $\text{Val}(X_{\mathcal{C}_i})$ de las variables $X_{\mathcal{C}_i}$ siguiendo (1). En la línea 5 se verifica si el número de variables del clique $\mathcal{C}_i$ es menor a dos, en tal caso el algoritmo no continua y toma el siguiente clique en $\mathcal{C}$. Esto

Algorithm 1: Context Separation Algorithm.

```
1  Procedure CS(C)
2      ξ ← ∅
3      foreach C_i ∈ C do
4          ξ ← ξ ∪ build-assignments(C_i)
5          if |C_i| < 2 then continue
6          foreach s, t ∈ C_i do
7              foreach W in the power set of C_i \ {s, t} do
8                  ξ ← ξ ∪ separation(ξ, s, t, W, ∅, ∅)
9      return ξ
10
11 Procedure separation(ξ, s, t, W, C, x)
12     if W \ C = ∅ then return ξ
13     C' ← W \ C
14     u ← first(C')
15     foreach x_u ∈ Val(X_u) do
16         ξ' ← ξ[x_u]
17         ξ'' ← ξ \ ξ'
18         if (X_s ⊥ X_t | X_{C'\u}, x ∪ x_u) then ξ' ← Split(ξ', s, t)
19         ξ ← ξ'' ∪ separation(ξ', s, t, W, C ∪ u, x ∪ x_u)
20     return ξ
21
```

es así porque para determinar independencias se necesita al menos dos variables. Luego, en las líneas 6 y 7, se seleccionan cuáles de las variables del clique $X_{\mathcal{C}_i}$ serán utilizadas para definir los contextos de las independencias que se ejecutarán durante el bucle interno. Esta selección es nuestra aproximación para obtener los contextos de las independencias y consiste en: para cada $s, t \in \mathcal{C}_i$ se obtiene el conjunto $W \subset \mathcal{C}_i$, donde W pertenece al *power set* de $\mathcal{C}_i \setminus (s, t)$. El costo de nuestra aproximación está en función del clique con mayor número de variables en G, denotemos a este número por $k^* \leq n$, entonces la aproximación tiene un costo de $O(2^{k^*-2})$ frente a los $O(2^{n-2})$ de no realizarla. Una justificación de esta aproximación se debe a las *Markov Properties* [7], hablando aproximadamente las dependencias que tiene una variable son más fuertes entre aquellas variables cuyos nodos son vecinos en la estructura G. Dado que un contexto implica una cierta dependencia entre las variables involucradas en la independencia contextual, entonces se puede suponer que la mayoría

de las CSIs ocurran entre variables cuyos nodos son próximos en G. La efectividad de esta suposición es corroborada por nuestros experimentos.

El bucle interno, realizado por la función `separation`, inicialmente recibe las asignaciones ξ del clique $\mathcal{C}_i$, los indices $s, t \in \mathcal{C}_i$ y el conjunto W. En base a estas evaluará, en línea 18, si las variables X_s y X_t son contextualmente independientes dado una asignación parcial a las variables X_W. Si la independencia se satisface en los datos, entonces se toma el subconjunto de ξ compatible con el contexto de la independencia y, como detalla la línea 18, se le aplica la operación de Split. Las asignaciones parciales para las variables X_W se construyen recursivamente en base al conjunto C y la asignación x, que llevan la pista de las variables y valores que han sido asignados en la recursión anterior. Por esto, inicialmente C y x son vacías. Por lo tanto, una asignación recursiva procede de la siguiente manera: en la línea 14 se toma la variable X_u donde $u \in W \setminus C$ y se itera en la línea 15 por sus asignaciones $x_u \in \text{Val}(X_u)$ haciendo una llamada recursiva en línea 19 con la asignación parcial $x \cup x_u$ y el nuevo conjunto $C \cup u$. Esto se detiene, en la línea 12, una vez que ya han sido asignados todos los valores posibles para las variables X_W.

5. Experimentación

Nuestros experimentos consisten en aprender modelos desde conjuntos de datos generados desde distribuciones subyacentes conocidas, Ising models binarios con la siguiente forma: $p(X) = \exp\{\sum_{s \in V} \theta_s \phi_s(X_s) + \sum_{(s,t) \in E} \theta_{st} \phi_{st}(X_{s \cup t}) - Z\}$. Donde $\phi_s(X_s) = 1$ si $x_s = 1$ y $\phi_{st}(X_{s \cup t}) = 1$ si $x_s = 1$ y $x_t = 1$. Los parámetros fueron elegidos aleatoriamente para generar distintas instancias del modelo. En todos los casos elegimos el parámetro del nodo s desde $\theta_s \sim \mathcal{N}(0, \sigma^2_{node})$. Para cada arista (s, t), establecimos su parámetro, independientemente de las restantes, como $\theta_{st} \sim |\mathcal{N}(0, \sigma^2_{edge})|$. Para todos los experimentos tuvimos en cuenta $\sigma_{node} = 0.5\ \sigma_{edge}$, y fuimos variando la elección de $\sigma_{edge} \in \{0.1, 0.3, 0.6, 0.9\}$. Mientras mayor es σ_{edge} menor es la fuerza de las dependencias en el modelo y su aprendizaje se torna más duro. Generamos diez Ising models diferentes por cada valor de σ_{edge} que muestreamos utilizando un Gibbs Sampler que descartó las primeras 10000 muestras (*burn-in phase*) y luego tomó 1000 muestras para armar un D.

Elegimos como competidor de CS al algoritmo HHC, uno de los algoritmos del estado del arte para aprender estructuras no contextualizadas desde conjuntos de datos [1]. Este algoritmo, al igual que CS, utiliza tests estadísticos durante el aprendizaje, de esta manera para ser justa la evaluación en ambos algoritmos utilizamos el test Pearson $\mathcal{X}^2$. Debido a que HHC fue diseñado para aprender modelos dirigidos, tuvimos que adaptarlo para nuestros experimentos. La adaptación consistió solamente en omitir su paso final en cual orienta las aristas de la estructura aprendida. Como explicamos en la sec. 3, el algoritmo CS necesita como entrada un conjunto de cliques $\mathcal{C}$ que representan a una estructura G. En nuestros experimentos utilizamos la estructura aprendida por HHC, obteniendo sus cliques utilizando el algoritmo Bron–Kerbosch.

Una vez obtenidas las estructura aprendidas por CS y HHC es necesario aprender sus parámetros para obtener un modelo, esto lo realizamos maximizando la función log-likelihood:

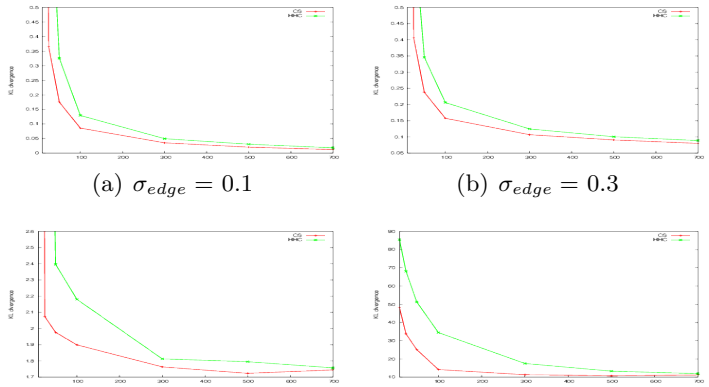
$$LL(\boldsymbol{\theta}) = \frac{1}{M}\sum_{m}^{M} \log p(D[m]) = \sum_{\alpha \in \mathcal{I}} \theta_\alpha \hat{P} - Z,$$

donde $\hat{P} := \frac{1}{M}\sum_{m}^{M} \phi_\alpha(D[m])$ es la suma de los valores del potencial sobre los datos, es decir, su promedio empírico en los datos. Tomando las derivadas de LL e igualándolas a cero encontramos que la solución θ^{ML} es *moment-matching*, es decir, satisface $\mathbb{E}_{\theta^{ML}}[\phi_\alpha] = \hat{P}$. Esto significa que los momentos de la distribución del modelo con parámetros θ^{ML} están emparejados con los promedios empíricos $\hat{P}$ de sus funciones indicador. Para esta optimización utilizamos L-BFGS.

Utilizamos como métrica para evaluar la calidad de los modelos aprendidos la *KL divergence* que tiene la forma:

$$KL(p \,||\, q) = \sum_{x \in \mathrm{Val}(X)} p(x) \log \frac{p(x)}{q(x)},$$

donde es igual a cero cuando $p = q$ y positiva en cualquier otro caso. Esta puede ser interpretada como una "distancia" de una distribución p



(a) $\sigma_{edge} = 0.1$ (b) $\sigma_{edge} = 0.3$

(c) $\sigma_{edge} = 0.6$ (d) $\sigma_{edge} = 0.9$

Figura 1. KL divergence para los modelos aprendidos desde conjuntos de datos por los algoritmos CS y HHC a medida que la cantidad de datos crece. Cada subfigura indica la dificultades del aprendizaje, mientras mayor σ_{edge} más duro es.

con respecto a otra distribución q en términos de su cantidad de información.

En la figura 5 mostramos los valores de KL promediados para 10 corridas sobre un número de datos creciente generados de grillas 4×4. Cada una de las subfiguras corresponde a diferentes escenarios de aprendizaje representados por los diferentes σ_{edge} que elegimos, siendo la esquina inferior derecha el escenario más duro. En todos los resultados podemos ver que para un reducido número de datos los modelos contextualizados resultan ser superiores a los no contextualizados y mientras los datos crecen ambos se tornan comparables. Los valores comparables de KL no ocurren por que se degrade la calidad de los modelos contextualizados, sino que se obtienen mejores modelos no contextualizados. La causa de estas mejoras se debe a la propiedad moment-matching del aprendizaje de parámetros. El aprendizaje de parámetros asigna parámetros cercanos a cero a aquellas funciones indicadores que tienen una baja frecuencia en los datos, al no tenerse en cuenta estas funciones en los modelos se obtiene un comportamiento similar al realizado por el algoritmo CS. Esta explicación es respaldada por el hecho de que los peores modelos no contextualizados son aprendidos para σ_{edge} grandes, es decir donde las independencias de

la distribución subyacente son débiles, necesitando el aprendizaje de parámetro una mayor cantidad de datos para hacer parámetros cercanos a cero.

6. Trabajo futuro

Hay muchas maneras en las cuales nuestro trabajo puede ser extendido. La principal es que nuestro algoritmo necesita de otro para obtener el conjunto inicial de cliques de una estructura, en base a los cuales aprende las independencias contextuales. En nuestros experimentos los cliques son suministrados por una versión modificada del algoritmo HHC. Uno de nuestros trabajos a futuro consistirá en extender las ideas expuestas de CS y diseñar un nuevo algoritmo que para un conjunto dado de datos aprenda las independencias contextualizadas para un modelo sin necesidad de suministrarle un conjunto inicial de cliques.

Referencias

1. Aliferis, C.F., Statnikov, A., Tsamardinos, I., Mani, S., Koutsoukos, X.: Local causal and markov blanket induction for causal discovery and feature selection for classification part i: Algorithms and empirical evaluation. JMLR 11, 171–234 (March 2010)
2. Amari, S.I.: Differential geometry of curved exponential families-curvatures and information loss. Annals of Statistics 10, 357–385 (1982)
3. Boutilier, C., Friedman, N., Goldszmidt, M., Koller, D.: Context-specific independence in bayesian networks. In: UAI. pp. 115–123 (1996)
4. Chickering, D.M., Heckerman, D., Meek, C.: A bayesian approach to learning bayesian networks with local structure. In: Uncertainty in Artificial Intelligence. pp. 80–89 (1997)
5. Fierens, D.: Context-specific independence in directed relational probabilistic models and its influence on the efficiency of gibbs sampling. In: European Conference on Artificial Intelligence. pp. 243–248 (2010)
6. Hammersley, J.M., Clifford, P.: Markov fields on finite graphs and lattices (1971)
7. Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann Publishers, Inc. (1988)
8. Poole, D., Zhang, N.L.: Exploiting contextual independence in probabilistic inference. J. Artif. Intell. Res. (JAIR) 18, 263–313 (2003)
9. Smith, J.E., Holtzman, S., Matheson, J.E.: Structuring conditional relationships in influence diagrams. Operations Research 41, 280–297 (1993)
10. Wexler, Y., Meek, C.: Inference for multiplicative models. In: Uncertainty in Artificial Intelligence. pp. 595–602 (2008)