Set de ejercicios 1

Predicción Estadística Estructurada

Iñaki Diez Lambies

Curso 2023 2024

Tabla de contenido

[Ejercicio 1 2](#_Toc154176357)

[Ejercicio 2 3](#_Toc154176358)

[Ejercicio 3 5](#_Toc154176359)

[Ejercicio 4 6](#_Toc154176360)

[Ejercicio 5 7](#_Toc154176361)

[Ejercicio 6 9](#_Toc154176362)

[Ejercicio 7 11](#_Toc154176363)

# Ejercicio 1

Diagrama

Descripción generada automáticamente

Given the previous Bayesian Network that defines the tree-augmented Naive Bayes classifier, provide the resulting factorization for the joint probability Dibujo de una persona

Descripción generada automáticamente con confianza media.

Un dibujo de un reloj

Descripción generada automáticamente con confianza media

# Ejercicio 2

Briefly explain why languages generated by a probabilistic grammar are not necessarily probabilistic and why not all probabilistic languages can be generated by a probabilistic grammar.

No todos los lenguajes probabilísticos pueden ser generados por una gramática probabilística. Esto se debe a que puede haber restricciones o patrones en la asignación de probabilidades a las cadenas de un lenguaje que una gramática probabilística no puede modelar.

El ejemplo presentado en el documento es el lenguaje:

Imagen que contiene objeto, reloj

Descripción generada automáticamente

donde la probabilidad de cada cadena en el lenguaje se define como:

Texto

Descripción generada automáticamente

Este lenguaje es un ejemplo clásico de un lenguaje que no puede ser generado por una gramática probabilística ya que la probabilidad asignada a cada cadena depende de manera no trivial del número de ocurrencias de 'a' y 'b', y está definida por la función de probabilidad​. Esta distribución de probabilidad es bastante compleja y no puede descomponerse en probabilidades de reglas individuales de una manera que una gramática probabilística pueda representar de manera adecuada. Las gramáticas probabilísticas estándar no pueden capturar fácilmente esta dependencia, ya que tratan las reglas de producción de forma independiente.

Por otro lado, una gramática probabilística se define como una gramática formal donde cada regla de producción tiene asociada una probabilidad. El propósito de estas probabilidades es modelar la "preferencia" o la frecuencia de uso de una regla sobre otras en ciertos contextos. La gramática genera cadenas a partir de estas reglas, con la probabilidad de generar una cadena específica siendo el producto de las probabilidades de las reglas utilizadas.

Cuando una gramática probabilística genera un lenguaje, está efectivamente creando un conjunto de cadenas con ciertas probabilidades asociadas. Sin embargo, esto no implica automáticamente que el lenguaje resultante sea un lenguaje probabilístico.

Un lenguaje probabilístico implica una distribución de probabilidad completa sobre todas las cadenas posibles dentro del lenguaje. Esto significa que, para cada cadena posible en el lenguaje, hay una probabilidad definida de que ocurra, y estas probabilidades deben sumar uno en total.

Supongamos que tenemos una gramática probabilística simple *G* que genera cadenas utilizando solo dos letras, 'a' y 'b'. Las reglas de producción con sus probabilidades asociadas podrían ser las siguientes:

1. *S* → *aS* (50% de probabilidad)
2. *S* → *bS* (30% de probabilidad)
3. *S* → *ϵ* (20% de probabilidad, donde *ϵ* denota la cadena vacía)

Esta gramática puede generar cadenas como 'a', 'b', 'aa', 'ab', 'ba', 'bb', 'aaa', etc. La probabilidad de generar una cadena específica es el producto de las probabilidades de las reglas utilizadas. A medida que las cadenas se vuelven más largas, la probabilidad de generar esas cadenas específicas disminuye (porque cada paso adicional multiplica la probabilidad total por un factor menor que uno). Esto podría implicar que las cadenas más largas tienen una probabilidad tan baja que se acercan a cero, pero aun así son parte del lenguaje.

En general, aunque esta gramática probabilística genera cadenas con ciertas probabilidades, no proporciona una distribución de probabilidad completa como en nuestro ejemplo para todas las posibles cadenas formadas por 'a' y 'b'. Las limitaciones en la generación de cadenas y la asignación de probabilidades indican que el lenguaje generado no es un lenguaje probabilístico en el sentido completo. Este ejemplo formal ayuda a entender la diferencia entre la generación de cadenas en una gramática probabilística y la definición completa de un lenguaje probabilístico.

# Ejercicio 3

Prove that the normalization factor of a CRF can be calculated as , with being the forward score of the string .

Para probar que el factor de normalización de un CRF puede ser calculado como se describe, es necesario entender primero lo que es el factor de normalización y el score forward de una cadena .

El factor de normalización se define como la suma de la exponencial de la suma ponderada de las funciones de características sobre todas las posibles secuencias de etiquetas , dadas las observaciones y los parámetros del modelo .

Aquí es el conjunto de todas las posibles secuencias de etiquetas, es la longitud de la secuencia, es el número de funciones de características, son las funciones de características, es la etiqueta en el tiempo , y son los parámetros del modelo asociados con la función de características .

Por otro lado, el *score forward* para un estado en el tiempo se calcula como la suma de la exponencial de la suma ponderada de las funciones de características sobre todas las posibles secuencias de etiquetas que terminan en el estado al tiempo .

O de forma recursiva:

Para probar que puede ser calculado utilizando , necesitamos demostrar que la suma de esta sobre todos los estados es igual al factor de normalización.

Dado que la suma interior recorre todas las posibles secuencias de etiquetas que terminan en el estado y la suma exterior suma sobre todos los posibles estados finales , juntos recorren todas las posibles secuencias de etiquetas en , lo que demuestra que la suma es igual a .

# Ejercicio 4

Given the following probabilistic transducers, T1 and T2, obtain the transducer resulting from the composition operation (T1 ◦ T2). The solution must leave a detailed record of all the steps to achieve it.

Diagrama

Descripción generada automáticamente

Diagrama

Descripción generada automáticamente

Diagrama

Descripción generada automáticamente

# Ejercicio 5

Suppose an instructor wants to determine whether a student has understood the class material based on the exam score. For this purpose, he has the following Bayesian network defined by the factorization associated with its joint probability.

P(I, H, U, E) = P(I) P(H) P(U | I, H) P(E | U),

Where:

* I represents whether the student is very intelligent (yes or no)
* H represents whether the student is very hardworking (yes or no)
* U represents whether the student has understood the class material (yes or no)
* E represents whether the student has obtained a high score on the exam (yes or no)

The probabilities are:

Tabla

Descripción generada automáticamente

What is the probability that a student who obtained a high exam score understood the class material?

Imagen que contiene Diagrama

Descripción generada automáticamente

Texto, Carta

Descripción generada automáticamente

Diagrama

Descripción generada automáticamente con confianza baja

# Ejercicio 6

Texto, Carta

Descripción generada automáticamente

Adaptar el algoritmo Forward para un CRF trigram implica ampliar el enfoque usado para el modelo bigram, incorporando la dependencia de un estado adicional previo. En el modelo bigram, las transiciones entre estados y las probabilidades asociadas solo consideran pares consecutivos de estados. Por otro lado, el modelo trigram introduce una dependencia más compleja que involucra tríos de estados consecutivos.

El algoritmo Forward para un CRF trigram se construye sobre la definición de una nueva función de transición , que ahora depende de los dos estados anteriores y del estado actual, además de la observación en el tiempo t. Esta función de transición se calcula mediante una exponencial de la suma ponderada de funciones de características específicas del modelo, donde los pesos son parámetros del modelo a aprender.

Al comenzar con el algoritmo, introducimos dos estados ficticios iniciales para manejar la dependencia tríada en el inicio de la secuencia. Estos estados ficticios se denotan comúnmente como , permitiendo que el algoritmo opere con una base coherente.

La recursión en el algoritmo Forward se adapta para acumular la probabilidad a través de todas las posibles secuencias de tres estados que pueden ocurrir hasta el tiempo . En el primer paso, inicializamos la matriz de probabilidades con la probabilidad de transición del estado inicial ficticio al primer estado real de la secuencia:

Para cada paso de tiempo subsiguiente , actualizamos las probabilidades acumuladas para cada posible combinación de estados y considerando todas las secuencias posibles que podrían haber llevado a esa combinación:

Este proceso se repite para cada tiempo desde 2 hasta , donde es el tiempo final de la secuencia.

La fórmula completa para la probabilidad de una secuencia de estados dados una secuencia de observaciones en el modelo CRF trigram es entonces la suma de las probabilidades acumuladas en el paso de tiempo final para todos los posibles estados finales:

Esta fórmula representa la probabilidad condicional de la secuencia completa de estados dado el modelo CRF trigram y la secuencia completa de observaciones, teniendo en cuenta las dependencias extendidas que el modelo trigram permite. La implementación efectiva de este algoritmo requiere una consideración cuidadosa de la eficiencia computacional, dada la suma adicional sobre los estados previos en cada paso de tiempo.

# Ejercicio 7

Texto, Carta

Descripción generada automáticamente

Imagen que contiene Dibujo de ingeniería

Descripción generada automáticamente

El algoritmo CKY adaptado para gramáticas de ramificación derecha calcula la probabilidad de una secuencia de entrada basándose en una versión simplificada de la técnica de programación dinámica. Para adaptar el algoritmo CKY para gramáticas de ramificación derecha, el pseudocódigo sería el siguiente:

1. Inicializar a 0 para todo .

2. Para cada de 1 a hacer:

Para cada A en N hacer:

3. Para cada longitud de span de 2 a hacer:

Para cada inicio de span de 1 a hacer:

Para cada posición de división de a hacer:

Para cada A en N hacer:

4. Devolver como la probabilidad de la secuencia.

El algoritmo utiliza una tabla tridimensional para almacenar las probabilidades máximas donde cada celda representa la probabilidad de que el segmento de la secuencia del índice i al j sea generado por el no terminal .

Se inicializa la tabla con las probabilidades de los no terminales que generan un terminal directamente. Luego, se rellena la tabla utilizando una regla de recurrencia que considera la probabilidad de descomposiciones en dos no terminales para segmentos de longitud creciente. La probabilidad final de la secuencia completa está en la celda , donde S es el símbolo de inicio.

El coste temporal del algoritmo se analiza en términos de las operaciones realizadas, que son proporcionales al número de combinaciones de segmentos de secuencia que se pueden formar. Dado que el proceso considera todas las posiciones de inicio, todas las posibles longitudes de los segmentos y todas las posiciones de división para cada longitud, el coste temporal es cúbico, es decir, , donde es la longitud de la secuencia de entrada.

Este coste se justifica por la necesidad de considerar todas las posibles particiones de la secuencia en dos partes para calcular la probabilidad de cada no terminal que genera esa secuencia. La especificidad de la ramificación derecha no reduce la complejidad asintótica, pero podría ofrecer mejoras prácticas en términos de constantes multiplicativas, ya que la estructura predictiva de la gramática limita el número de producciones posibles a considerar.