Modelos Descriptivos y Predictivos II

Ernesto Martínez Gómez

Índice general

1 Tema 1: Introducción al machine learning

1.1 Métodos de partición de datos

Al usar aprendizaje automático, los datos se pueden dividir de muchas formas en entrenamiento y test. Algunas de las más comunes son:

1. **Resustitución:**
   - Todos los datos para entrenar y validar.
   - Inconveniente: es muy optimista, ya que el modelo se ajusta a los datos con los los que se testea.
2. **Partición:**
   - Se crea un subconjunto de datos para entrenar y otro para testear.
   - Se puede hacer partición aleatoria o estratificada.
   - Inconveniente: se desaprovechan datos.
3. **Validación cruzada en k bloques** (k-fold cross-validation):
   - Se divide en k bloques y se entrena con k-1 bloques y se testea con el restante.
   - Se repite k veces.
   - Se promedian los resultados.
   - Inconveniente: es costoso computacionalmente y se entrena con menos datos.
4. **Exclusión individual** (Leaving-one-out):
   - Cada dato individual se usa para testear un sistema entrenado con el resto (n-1) de individuos.
   - Equivale a k-fold con k=n.
   - Inconveniente: es muy costoso computacionalmente.

1.1.1 Métodos de partición gráficamente

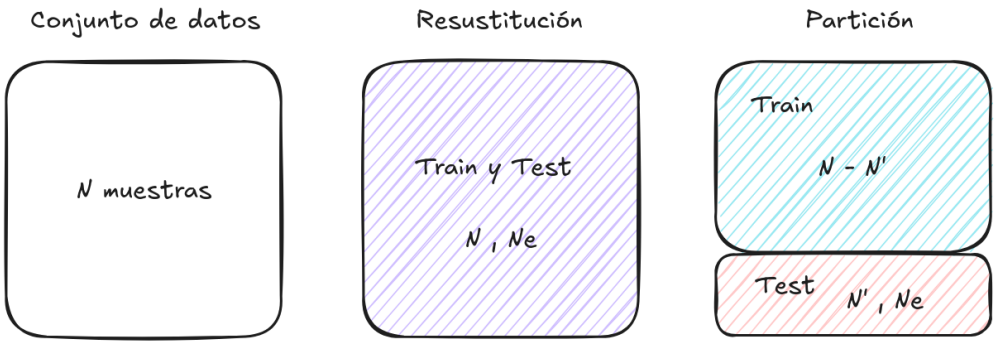

Figura 1.1: Partición de datos por resustitución y partición

En las imágenes, N es el número de datos, N_e es el número de errores, N' es el número de datos de testeo y N'_e es el número de errores en test.

Para el método de partición, la tasa de errores se calcula como $\frac{N_e}{N}$ y la talla de entrenamiento efectiva es N.

En el método de partición, la tasa de errores se calcula como $\frac{N'_e}{N'}$ y la talla de entrenamiento efectiva es $N - N'$.

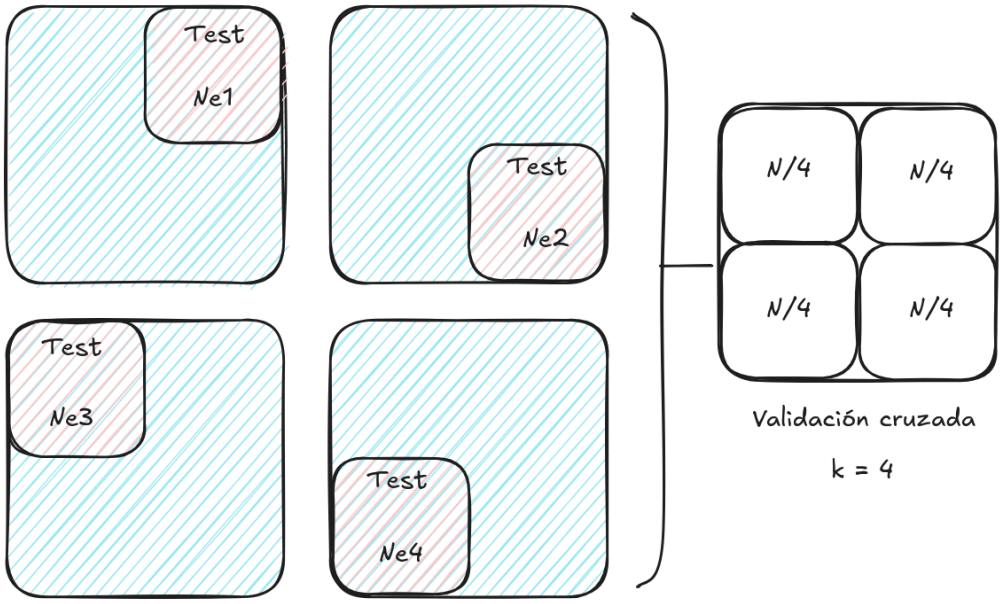

Figura 1.2: Validación cruzada con k = 4

Para la validación cruzada, la tasa de errores se calcula como $\frac{N_{e1}+N_{e2}+N_{e3}+N_{e4}}{N}$ y la

talla de entrenamiento efectiva es $\frac{3N}{4}$.

De forma general, la tasa de errores se calcula como $\frac{1}{k}\sum_{i=1}^{k} N_{ei}$ y la talla de entrenamiento efectiva es $\frac{k-1}{k}N$.

2 Tema 2: Optimización matemática

2.1 Clasificación, regresión y optimización

Podemos definir un modelo como una función parametrizada por Θ, que es un conjunto de parámetros.

$$\mathscr{F} = \{f_\Theta : X \to Y, \quad \Theta \in \mathbb{R}^D\}$$

Donde X es el espacio de entrada y Y el espacio de salida.

En el caso de la clasificación, Y es un conjunto finito de etiquetas

$$Y = \{1, 2, \dots, C\}$$

En el caso de la regresión, Y es un conjunto continuo.

$$Y \equiv \mathbb{R}$$

Y típicamente, $X \equiv \mathbb{R}^d$

De esta forma, encontramos dos etapas en el ciclo de vida de un modelo:

1. Entrenamiento: Ajustar los parámetros del modelo dado un conjunto de datos de entrenamiento.

   $$\text{Dado } S \subset X \times Y \to \text{Estimar } \Theta \in \mathbb{R}^D$$

2. Evaluación: Medir la calidad del modelo en un conjunto de datos de prueba.

   $$\text{Dados } \Theta yx \in X \to \text{Estimar } y \in Y = f_\Theta(x)$$

3 Tema 3: Redes neuronales

3.1 Algoritmo backpropagation

Supongamos una red neuronal donde existen L capas de M_l neuronas cada una. Esta red neuronal sirve para solucionar un problema de regresión usando J variables como regresoras, también llamadas características o *features* para predecir K variables.

El entrenamiento de esta se compone de dos partes en cada *epoch*:

- *Forward pass*: Se calcula la predicción y el error para los parámetros actuales.
- *Backward pass*: Se calculan los nuevos pesos corregidos para disminuir el error.

Se va a utilizar a partir de ahora notación vectorial con el fin de simplificar las fórmulas y cálculos. Es decir, w será interpretado como un vector y W como una matriz.

3.1.1 Variables

Vamos a encontrar las siguientes variables:

- X: Matriz con los datos de entrenamiento de tamaño $N \times J$ (Individuos $\times$ Características). Se usará x_n como vector que representa a un individuo.
- y: Matriz con el valor real de las variables a predecir o *target* para cada individuo. Es de tamaño $N \times K$.
- $W^{(l)}$: Matriz de pesos de tamaño $M_l \times M_{l-1}$. Cada elemento $w_{ij}^{(l)}$ corresponde al peso de la conexión entre la neurona i de la capa l con la neurona j de la capa $l-1$.
- $z^{(l)}$: Vector con los valores de cada neurona de la capa l.
- $g^l(z)$: Función de activación de la capa l.
- $s^{(l)}$: Valor de cada neurona de la capa l tras aplicar la función de activación $g^l(z^{(l)})$

3.1.2 *Forward pass*

Para cada capa, se debe calcular el valor de todas las neuronas e ir propagando los valores hacia delante en la red neuronal hasta llegar a la última capa.

El valor de las neuronas de una capa l se calcula de la siguiente forma:

$$s^{(l)} = g^l(z^{(l)})$$

Donde:

$$z^{(l)} = W^{(l)} \cdot s^{(l-1)} + b^{(l)}$$

En el caso de la capa de entrada ($l = 0$), $s_0 = x_n$

Ejemplo: Cálculo de la primera iteración de un *forward pass*

Suponemos una red neuronal con 3 capas: entrada, salida y una oculta en medio, de 4, 2 y 1 neurona respectivamente.
Los valores para iniciar los pesos de la red neuronal $W^{(l)}$ son un vector de unos para todas las capas. Lo mismo para los vectores de sesgo $b^{(l)}$.

3.1.3 *Backward pass*

La forma de entrenar la red neuronal es ir actualizando los pesos en cada iteración para acercar las predicciones a los valores reales de las variables a predecir.

Para esto, hay que definir una función de coste. Usaremos el Error Cuadrático Medio (MSE o *Mean Squared Error*)

$$q = \frac{1}{N} \sum_{n=1}^{N} q_n$$

Donde q_n es el error del individuo n:

$$q_n = \frac{1}{2} \sum_{k=1}^{K} \left(y_k - s_k^{(l)} \right)^2$$

Donde:

- y_k es el valor real de la variable k para el individuo n
- $s_k^{(l)}$ es el valor estimado de la variable k para el individuo n

Ahora que tenemos definida la función de coste podemos calcular el error de la red neuronal en un momento t y reajustar los valores de los parámetros.

La forma de calcular los nuevos valores de los parámetros es mediante el descenso por gradiente o *gradient descent*. Se calcula la derivada de la función de error o coste y se da un paso en la dirección que reduzca el error.

El algoritmo ***backpropagation*** entra en juego a la hora de enviar el error a las neuronas anteriores a la última capa con el objetivo de recalcular los valores de sus parámetros. Se basa en las derivadas parciales y la regla de la cadena para propagar el error a lo largo de la red.

Para ello primero debemos definir las derivadas parciales en función de cada parámetro.

Supongamos que nuestra función de activación es la función sigmoide:

$$g(x) = \frac{1}{1 + e^{-x}}$$

4 Tema 6: Support Vector Machines

4.1 Clasificación en 2 clases con Funciones Discriminantes Lineales

Si usamos dos clases, es muy útil etiquetarlas como -1 y 1 y usar un único vector de pesos $\theta = \theta_1 - \theta_2$. De esta forma clasificamos de la siguiente forma:

- Clase 1 o +1: $\phi_1(x) \geq \phi_2(x) \Rightarrow \theta_1^t x \geq \theta_2^t x \Rightarrow \theta^t x \geq 0$
- Clase 2 o -1: $\phi_1(x) < \phi_2(x) \Rightarrow \theta_1^t x < \theta_2^t x \Rightarrow \theta^t x < 0$

Clasificador, $f_\theta(x) : \mathbb{R}^n \to \{-1, 1\}$

$$f_\theta(x) = \begin{cases} 1 & \text{si } \theta^t x \geq 0 \\ -1 & \text{si } \theta^t x < 0 \end{cases}$$

Se define la función discriminante lineal (FDL) como

$$\phi : \mathbb{R}^d \to \mathbb{R} \text{ tal que } \phi(x; \Theta) = \theta^t x + \theta_0 = \sum_{i=1}^{d} \theta_i x_i + \theta_0$$

Donde $\Theta = (\theta, \theta_0) : \theta \in \mathbb{R}^d$ es el vector de pesos y $\theta_0 \in \mathbb{R}$ se denomina umbral. El número de parámetros de Θ es $D = d + 1$.

En forma compacta:

$$c_n \cdot (\theta^t x_n + \theta_0) \geq 0 \quad 1 \leq n \leq N$$

Donde $c_n \in \{-1, 1\}$ es la clase de la observación x_n.

Frontera de decisión: $F = x : \theta^t x = 0$

Propiedad: La distancia de un punto x a la frontera de decisión (r_x) es proporcional

al valor absoluto de $\theta^t x$: $r_x \propto |\theta^t x|$

4.2 Propiedades de las Funciones Discriminantes Lineales

1. Una FDL, $\phi(x;\Theta)$, define un hiperplano de separación $H = x : \theta^t x + \theta_0 = 0$.
2. H divide $\mathbb{R}^d$ en dos semiespacios: $H^+ = x : \theta^t x + \theta_0 > 0, c = +1$ y $H^- = x : \theta^t x + \theta_0 < 0, c = -1$.
3. Dado un $\gamma \in \mathbb{R}^+$, entonces $\gamma \cdot \phi(x;\Theta)$ representa el mismo hiperplano H.
4. El vector de pesos θ es ortogonal a H y su vector unitario es $\frac{\theta}{||\theta||}$.
5. La distancia de cualquier x_s a H es:

$$
\begin{aligned}
x_s &= r_{x_s}\frac{\theta}{\|\theta\|} + x_o, \\
\theta^t x_s &= r_{x_s}\frac{\theta^t\theta}{\|\theta\|} + \theta^t x_o, \\
\theta^t x_s + \theta_0 &= r_{x_s}\frac{\|\theta\|\|\theta\|}{\|\theta\|} + \underbrace{\theta^t x_o + \theta_0}_{(x_o \in H) \Longrightarrow 0}, \\
r_{x_s} &= \frac{\left|\phi(x_s;\Theta)\right|}{\|\theta\|} = \frac{\left|\theta^t x_s + \theta_0\right|}{\|\theta\|}.
\end{aligned}
$$

Donde x_o es un punto de H.

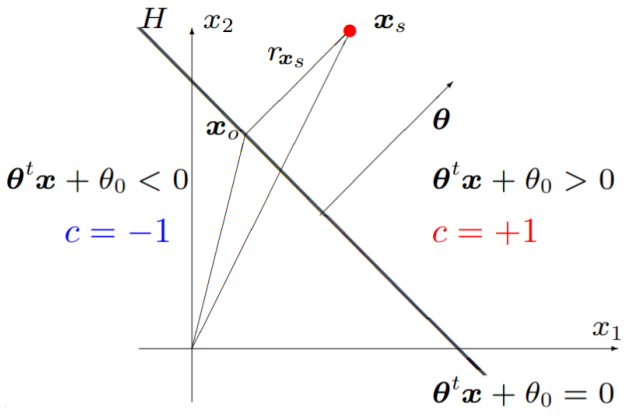

Figura 4.1: Distancia de un punto a la frontera de decisión.

4.3 Hiperplano y márgen de un Hiperplano separador

- Para un conjunto de muestras linealmente separables existen infinitos hiperplanos separadores.
- El margen τ de un hiperplano H es la mínima distancia entre H y la muestra más cercana a cualquiera de sus clases.

$$\tau = \min_n \left| \frac{\theta^t x_n + \theta_0}{\|\theta\|} \right|$$

- Un hiperplano H es óptimo si maximiza el margen τ.

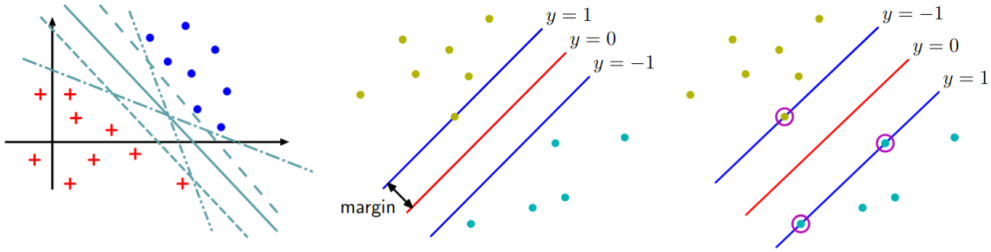

Figura 4.2: Hiperplano cualquiera vs Hiperplano óptimo.

La distancia de caulquier punto x_n a H será siempre mayor que el margen τ.

Las muestras de la frontera que delimita el margen (a ambos lados de H) se denominan **vectores soporte**. Cumplen que:

$$\frac{|\phi(x_i;\Theta)|}{\|\theta\|} = \tau \quad \forall x_i \in \mathscr{V}$$

siendo $\mathscr{V}$ el conjunto de todos los vectores soporte.

4.3.1 Propiedades de los clasificadores de orden máximo

Voy a saltarme algunas partes para igualar la clase de hoy

4.4 Clase 22 nov. Máquinas de soporte y núcleos

4.4.1 Calcular vector de pesos

θ^*

Márgenes duros

$$\theta^* = \sum_{n=1}^{N} c_n x_n$$

$$\theta_0^* = c_m - \theta^{*t} x_m, \quad \alpha_m > 0$$

$$\theta_0^* = c_m - \sum_{n \in \mathscr{V}} c_n \alpha_n \mathscr{K}(x_n, x_m), \quad \alpha_m > 0$$

4.4.2 Márgenes blandos

$$\theta^* = \sum_{n=1}^{N} c_n \alpha_n x_n$$

Ejercicio 1 GitHub

Dado el siguiente conjunto de datos y un Kernel polinómico definido como

$$K(x, y) = (\langle x, y \rangle + 1)^2$$
$$\langle x, y \rangle = x^t \cdot y = x_1 y_1 + x_2 y_2$$

Y las muestras:
$(x_1 = [1, 1], c_1 = +1)$
$(x_2 = [0, 0], c_2 = +1)$
$(x_3 = [0, 1], c_3 = -1)$
$(x_4 = [1, 0], c_4 = -1)$
Calcula la matriz Kernel para dichas muestras y resuelve el problema de clasificación con márgenes duros.

Tenemmos que ir calculano uno a uno los elementos de la matriz Kernel aplicando la función Kernel a cada par de muestras.

- $K(x_1, x_1) = (x_1^t \cdot x_1 + 1)^2 = (2 + 1)^2 = 9$

- $K(x_1, x_2) = (x_1^t \cdot x_2 + 1)^2 = (0 + 1)^2 = 1$
- $K(x_1, x_3) = (x_1^t \cdot x_3 + 1)^2 = (1 + 1)^2 = 4$
- $K(x_1, x_4) = (x_1^t \cdot x_4 + 1)^2 = (1 + 1)^2 = 4$
- $K(x_2, x_2) = (x_2^t \cdot x_2 + 1)^2 = (0 + 1)^2 = 1$
- $K(x_2, x_3) = (x_2^t \cdot x_3 + 1)^2 = (0 + 1)^2 = 1$
- $K(x_2, x_4) = (x_2^t \cdot x_4 + 1)^2 = (0 + 1)^2 = 1$
- $K(x_3, x_3) = (x_3^t \cdot x_3 + 1)^2 = (1 + 1)^2 = 4$
- $K(x_3, x_4) = (x_3^t \cdot x_4 + 1)^2 = (0 + 1)^2 = 1$
- $K(x_4, x_4) = (x_4^t \cdot x_4 + 1)^2 = (1 + 1)^2 = 4$

La matriz Kernel resultante es:

$$K = \begin{bmatrix} 9 & 1 & 4 & 4 \\ 1 & 1 & 1 & 1 \\ 4 & 1 & 4 & 1 \\ 4 & 1 & 1 & 4 \end{bmatrix}$$

Resolvemos el problema de clasificación con márgenes duros (no hay C)

```python
from scipy.optimize import minimize
import numpy as np

# Datos y etiquetas
x = np.array([[1, 1], [0, 0], [0, 1], [1, 0]])
labels = np.array([1, 1, -1, -1])
n_samples = len(labels)

# Definición del kernel polinómico
def polynomial_kernel(x, y):
    return (np.dot(x, y) + 1) ** 2



# Construcción de la matriz kernel
kernel_matrix = np.zeros((4, 4))
for i in range(4):
    for j in range(4):
        kernel_matrix[i, j] = polynomial_kernel(x[i], x[j])

```

```
21
22  # Definir la función objetivo para la optimización dual de SVM
23  def objective(alphas):
24      return 0.5 * np.sum([alphas[i] * alphas[j] * labels[i] * labels[
        j] * kernel_matrix[i, j] for i in range(n_samples) for j in range
        (n_samples)]) - np.sum(alphas)
25
26  # Restricción: \sum \alpha_i y_i = 0
27  def constraint_eq(alphas):
28      return np.dot(alphas, labels)
29
30  # Restricción: \alpha_i ≥ 0
31  bounds = [(0, None) for _ in range(n_samples)]
32
33  # Punto inicial para la optimización
34  initial_alphas = np.zeros(n_samples)
35
36  # Resolver el problema de optimización
37  result = minimize(
38      fun=objective,
39      x0=initial_alphas,
40      bounds=bounds,
41      constraints={'type': 'eq', 'fun': constraint_eq},
42      method='SLSQP'
43  )
44
45  # Extraer los valores de \alpha
46  optimal_alphas = result.x
47
48  print(optimal_alphas)
```

Clasificar las muestras de entrenamiento

$$f(x_i) = \sum_{n=1}^{N} c_n \alpha_n K(x_n, x_i) + b$$

El sesgo (bias b) se puede calcular como:

$$b = c_n - \sum_{m \in \mathscr{V}} c_m \alpha_m K(x_n, x_m)$$

Calculamos el sesgo

Por ejemplo, para la muestra $x_1, c_1 = +1$:

$$
\begin{aligned}
b &= \sum_{m=1}^{4} c_m \alpha_m K(x_m, x_1) \\
&= +1 - (2 * 9 + 1 * (10/3) - 4 * (8/3) - 4 * (8/3)) \\
&= 1 - (54/3 + 10/3 - 32/3 - 32/3) \\
&= 1 - 0 = 1
\end{aligned}
$$

5 Tema 7: Predicción estructurada. Modelos ocultos de Markov

5.1 Introducción

5.2 Aplicaciones

- Natural Language Processing (NLP)
  - Part-of-speech tagging
  - Parsing
  - Machine translation
  - Information extraction
- Procesamiento de imágenes
  - Reconocimiento de caracteres
  - Análisis visual de escenas
- Procesamiento de voz
  - Transcripción automática
  - Texto a voz
- Bioinformática
  - Predicción de la estructura de proteínas
- Robótica
  - Localización y mapeo simultáneos (SLAM)
  - Planificación de trayectorias

5.3 Espacio de búsqueda

$\mathcal{Y}$ es potencialmente infinito, por lo que $f(x) = y \in \mathcal{Y}$ será intratable.

5.4 Lenguajes: Definiciones básicas

5.5 Modelos aceptores: Autómatas

5.5.1 Autómata finito determinista (DFA)

Es una tupla $A = (\Sigma, Q, \delta, q_0, F)$ donde:

- Σ es el alfabeto de entrada
- Q es el conjunto finito y no vacío de estados del autómata
- δ es la función de transición entre estados $Q \times \Sigma \rightarrow Q$
- $q_0 \in Q$ es el estado inicial
- $F \subseteq Q$ es el conjunto de estados finales

δ es una matriz de transición de tamaño $|Q| \times |\Sigma|$. Cada fila de la matriz representa un estado y cada columna representa un símbolo del alfabeto. Cada celda contiene el estado al que se transita desde el estado correspondiente al número de fila al leer el símbolo correspondiente al número de columna.

Ejemplo de autómata

$$A = (\Sigma = a, b, Q = 0, 1, 2, \delta, q_0 = 0, F = 2)$$

Para el ejemplo anterior, la matriz de transición sería:

$$\begin{bmatrix} 1 & 0 \\ 2 & 1 \\ 2 & 2 \end{bmatrix}$$

5.6 Equivalencia de modelos

Equivalencia de modelos

Dos modelos, M_1 y M_2, son equivalentes si definen (aceptan o generan) el mismo lenguaje.

$$M_1 \equiv M_2 \iff L(M_1) = L(M_2)$$

Teorema. Dada una gramática regular, G, se puede construir un autómata finito determinista, A, tal que $L(G) = L(A)$.

5.7 Incertidumbre

5.8 Lenguaje probabilístico

Lenguaje $L \subseteq \Sigma^*$, dado un alfabeto Σ.

Lenguaje generado por una gramática $L \subseteq \Sigma^*$, dado un alfabeto Σ y una gramática $G = (N, \Sigma S, \mathscr{P})$.

$$L(G) = \{x | x \in \Sigma^* : S \Rightarrow^+ x\}$$

5.9 Gramáticas Regulares Probabilísticas

Se expresa como $G_\theta = (G, P)$, donde

- $G = (N, \Sigma, S, \mathscr{P})$ es una gramática regular característica
- $P : \mathscr{P} \to [0, 1]$ es una función de probabilidad

5.10 Ejercicios

5.10.1 Ejercicio 1

Ejercicio 1

$$G = (N = \{S, A, B\}, \Sigma = \{a, b\}, S, P)$$

$$P = \begin{cases} S \to aA \mid bB \mid b \\ A \to aA \mid bB \mid b \\ B \to bB \mid b \end{cases}$$

Apartado 1

Sí que es una gramática regular, ya que todas las producciones son de la forma $A \to aB$ o $A \to a$, donde A y B son no terminales y a es un terminal.

Apartado 2

Primero: El primer símbolo de la secuencia es un no terminal B, por lo que no se puede haber generado con esta gramática.

Segundo: No se puede generar la secuencia bA con esta gramática.

Tercero: Se podría generar a través de la producción $S \to aA \quad A \to bB$.

Cuarto: Se podría generar a través de la producción $S \to bB \quad B \to b$.

Apartado 3

Primero: Se genera a partir de la siguiente secuencia de producciones:

$$S \to aA \to aaA \to aaaA \to aaabB \to aaabb$$

Segundo: No se podría generar, ya que no se puede generar una b después de una a.

Tercero: Se genera a partir de la siguiente secuencia de producciones:

$$S \to aA \to ab$$

Cuarto: No se puede generar por lo mismo que en el segundo apartado.

5.10.2 Ejercicio 2

Ejercicio 2

1. ¿Cuál es la longitud mínima de la cadena de símbolos que puede aceptar?
2. ¿Cuáles serían las cadenas de mínima longitud que puede aceptar?
3. ¿Dicho autómata aceptaría las cadenas del tipo $a^n b^m c^n$, con $n > 0$ y $m > 0$?
4. ¿Cuál sería la gramática regular equivalente?

Apartado 1

La longitud mínima es 1, ya que se puede llegar a un estado final (2 o 3) desde el estado inicial.

Apartado 2

Las cadenas de mínima longitud que puede aceptar son b y c.

Apartado 3

El autómata sí podría aceptar cadenas del tipo $a^n b^m c^n$, ya que se puede repetir el símbolo a y el símbolo b y el símbolo c tantas veces como se quiera en la misma cadena.

Apartado 4

La gramática regular equivalente sería:

$$P = \begin{cases} S \to aS \mid bA \mid b \mid cB \mid c \\ A \to bA \mid b \mid cB \mid c \\ B \to cB \mid c \end{cases}$$

Donde S es el estado inicial (1), A es el estado final 2 y B es el estado final 3.

5.10.3 Ejercicio 3

Ejercicio 3

Dada la gramática regular probabilística, ¿cuál será la probabilidad de la interpretación más probable para la cadena de entrada $a\,b\,a\,b$?

$$\begin{array}{llll} S \to aS & 0.3 & | & A \to bA \quad 0.2 \\ S \to bS & 0.2 & | & A \to aA \quad 0.2 \\ S \to bA & 0.3 & | & A \to aS \quad 0.1 \\ S \to b & 0.2 & | & A \to b \quad 0.5 \end{array}$$

Forma 1:

$$S \xrightarrow{0.3} aS \xrightarrow{0.2} abS \xrightarrow{0.3} abaS \xrightarrow{0.2} abab$$

Probabilidad: $0.3 \cdot 0.2 \cdot 0.3 \cdot 0.2 = 0.0036$

Forma 2:

$$S \xrightarrow{0.3} aS \xrightarrow{0.3} abA \xrightarrow{0.2} abaA \xrightarrow{0.5} abab$$

Probabilidad: $0.3 \cdot 0.3 \cdot 0.2 \cdot 0.5 = 0.009$

Forma 3:

$$S \xrightarrow{0.3} aS \xrightarrow{0.3} abA \xrightarrow{0.1} abaS \xrightarrow{0.2} abab$$

Probabilidad: $0.3 \cdot 0.3 \cdot 0.1 \cdot 0.2 = 0.0018$

5.10.4 Ejercicio 4. Ejercicio de examen

Probabilidad de transición:

Prob. Transición	1	2	3	F
1	0.8	0.2	0.0	0.0
2	0.0	0.6	0.3	0.1
3	0.4	0.0	0.5	0.1

Probabilidad de emisión:

Prob. Emisión	a	b
1	0.3	0.7
2	0.0	1.0
3	0.6	0.4

Ejercicio examen

a) Representad gráficamente este modelo.

b) Calculad la probabilidad de que el modelo genere la cadena `abbba` a través de la secuencia de estados 31223.

c) Completad la traza del algoritmo de Viterbi para obtener la secuencia de estados más probable con la que el modelo da cuenta de la cadena `abb`. Obtened además la probabilidad resultante (probabilidad de Viterbi).

d) Estimad los parámetros de M mediante una iteración de estimación por Viterbi, a partir de los siguientes pares (cadena, secuencia óptima de estados):

cadenas	`abbba`	`bbba`	`bba`
estados	31223	1223	333

Además, también se debe añadir la cadena `abb` y la secuencia óptima de estados obtenida en el apartado anterior (total 4 pares).

Apartado a

Está dibujado en el ipad.

Apartado b

Para obtener la cadena `abbba` usando viterbi a través de `31223`, se puede hacer de la siguiente forma:

1. $I \to 3$ Probabilidad(a) $= 0.2 \cdot 0.6 = 0.12$
2. $3 \to 1$ Probabilidad(b) $= 0.12 \cdot 0.4 \cdot 0.7 = 0.0336$
3. $1 \to 2$ Probabilidad(b) $= 0.0336 \cdot 0.2 \cdot 1 = 0.00672$
4. $2 \to 2$ Probabilidad(b) $= 0.00672 \cdot 0.6 \cdot 1 = 0.004032$
5. $2 \to 3$ Probabilidad(b) $= 0.004032 \cdot 0.3 \cdot 0.4 = 0.0012096$

6. $3 \to F$ Probabilidad(b) $= 0.0012096 \cdot 0.1 = 0.00012096$

Resultado = 0.000217728

Apartado c

	a	b		
1	0.12	$1 : 0.0672$ $3 : 0.1008$		
2	$\varnothing$	$1 : 0.0240$ $2 : \varnothing$		
3	0.36	$2 : \varnothing$ $3 : 0.072$		
F				

Apartado d

cadenas	`abbba`	`bbba`	`bba`	`abb`
estados	$31223F$	$1223F$	$333F$	$321F$

- Veces que se transita del estado 1 al estado 1: 0
- Veces que se transita del estado 1 al estado 2: 1
- Veces que se transita del estado 1 al estado 3: 0
- Veces que se transita del estado 1 al estado F: 1
- Veces que se transita del estado 2 al estado 1: 0
- Veces que se transita del estado 2 al estado 2: 2
- Veces que se transita del estado 2 al estado 3: 1

6 Modelos gráficos

Los nodos representan variables aleatorias

El grafo representa un conjunto de dependencias y factoriza una distribución.

6.1 Redes bayesianas

Red Bayesiana

Una Red Bayesiana es un modelo gráfico que representa un conjunto de variables aleatorias y sus dependencias condicionales a través de un grafo acíclico dirigido (DAG).

Nos da una forma de calcular la probabilidad conjunta

$$P(a, b, c) = P(a)P(b)P(c|a, b)$$

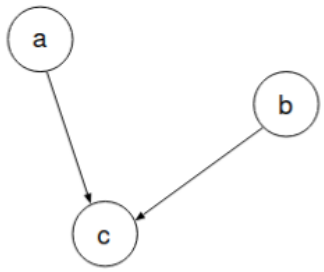

Figura 6.1: Ejemplo de red bayesiana

6.1.1 Idependencia condicional

Se dice que a es incondicionalmente independiente de b, y se denota como $a \perp b$, si y solo si

$$P(a, b) = P(a)P(b)$$

$$P(a|b) = P(a)$$

Se dice que a es condicionalmente independiente de b dado c, y se denota como $a \perp b|c$, si y solo si

$$P(a, b|c) = P(a|c)P(b|c)$$
$$P(a|b, c) = P(a|c)$$

Reglas de independencia condicional e incondicional

Dirección causal: $(a \perp b|c) \quad (a \not\perp b)$

$$P(a, b|c) = \frac{P(a, b, c)}{P(c)} = \frac{P(a|c)P(b|c)P(c)}{P(c)} = P(a|c)P(b|c)$$

pero en general $P(a, b) \neq P(a)P(b)$

Causa común: $(a \perp b|c) \quad (a \not\perp b)$

$$P(a, b|c) = \frac{P(a, b, c)}{P(c)} = \frac{P(a|c)P(b|c)P(c)}{P(c)} = P(a|c)P(b|c)$$

pero en general $P(a, b) \neq P(a)P(b)$

Estructura en V: $(a \not\perp b) \quad (a \perp b|c)$

$$P(a, b|c) \neq P(a|c)P(b|c)$$

pero

$$P(a, b) = \sum_c P(a, b, c) = \sum_c P(a)P(b)P(c|a, b) = P(a)P(b)$$

6.1.2 Inferencia con Redes Bayesianas

En general, el problema consiste en calcular la probabilidad a posteriori de alguna variable x a partir de las distribuciones conjuntas asociadas a una red bayesiana,

dada alguna evidencia e (como valores de otras variables) y sin importar el resto de las variables f.

$$P(x \mid e) = \frac{P(x,e)}{P(e)} \quad \text{con:} \quad P(x,e) = \sum_f P(x,e,f); \quad P(e) = \sum_{x,f} P(x,e,f)$$

El objetivo es calcular eficientemente $P(x,e) y P(e)$.

Ejercicio ejemplo

Ejercicio de la página 14 de las diapositivas.

$A = f, L = n, C = m$

$$P(A = f|L = n, C = m) = \frac{P(A = f, L = n, C = m)}{P(L = n, C = m)} =$$

$$\frac{P(L = n)P(A = f|L = n)P(C = n|L = n, A = f)}{\sum_{a=\{p,f\}} P(A = a, L = n, C = m)} =$$

$$\frac{0.8 \cdot 0.4 \cdot 0.9}{0.288 + (0.8 \cdot 0.6 \cdot 0.01)} = \frac{0.288}{0.288 + 0.0048} = \frac{0.288}{0.2948} = 0.9836$$

Apartado B del mismo ejemplo

No llueve, cuál es la mejor predicción sobre el estado del césped?

$$\text{argmax}_{c\in\{r,m\}} P(C = c|L = n)$$

$$P(C = c|L = n) = \frac{P(C = r, L = n)}{P(L = n)}$$

Como $P(L = n)$ no depende de c, podemos ignorarla si queremos saber el máximo.

$$P(C = r, L = n) = \sum_{a\in\{p,f\}} P(C = r, L = n, A = a)$$

Para $c = r$:

$$P(C = r, L = n, A = f) + P(C = r, L = n, A = p) =$$

$$P(L = n)P(A = p|L = n)P(C = r|L = n, A = p) + P(L = n)P(A = f|L = n)P(C = r|L = n, A = f) =$$

$$(0.8 \cdot 0.6 \cdot 0.99) + (0.8 \cdot 0.4 \cdot 0.1) = 0.4752 + 0.032 = 0.5072$$

Para calcular la probabilidad, dividimos por $P(L = n)$

$$P(C = r|L = n) = \frac{0.5072}{0.8} = 0.634$$

Como solo hay dos casos, $P(C = m|L = n) = 1 - 0.634 = 0.366$

Por lo tanto, la mejor predicción es que el césped esté reseco.

Ejemplo del cáncer

¿Cuál es la probabilidad de que un paciente no fumador no tenga cáncer si la radiografía ha dado un resultado negativo pero sufre de disnea?

Nos piden $P(C = n|F = n, X = n, D = s)$

$$P(C = n|F = n, X = n, D = s) = \frac{P(C = n, F = n, X = n, D = s)}{P(F = n, X = n, D = s)} =$$

$$\frac{\sum_{p \in \{b,a\}} P(C = n, F = n, X = n, D = s, P = p)}{\sum_{c \in \{n,p\}} \sum_{p \in \{b,a\}} P(C = c, F = n, X = n, D = s, P = p)}$$

Para simplificar, la forma de calcular $P(C, F, X, D, P)$ es

$$P(C, F, X, D, P) = P(P) \cdot P(F) \cdot P(C|P, F) \cdot P(X|C) \cdot P(D|C)$$

Ejercicio de examen sobre el cáncer

¿Cuál es la mejor predicción sobre el cáncer si el paciente es fumador, sufre disnea y el resultado de la radiografía ha sido dudoso? Proporcione la probabilidad de cada una.

Nos piden argmax$c \in \{n, s\} P(C = c|F = f, X = d, D = s)$

$$P(C = c|F = f, X = d, D = s) = \frac{P(C = c, F = f, X = d, D = s)}{P(F = f, X = d, D = s)} =$$

$$\frac{\sum_{p\in\{b,a\}} P(C = c, F = f, X = d, D = s, P = p)}{\sum_{c\in\{n,p\}} \sum_{p\in\{b,a\}} P(C = c, F = f, X = d, D = s, P = p)}$$

Podemos simplificar la fórmula quitando el denominador, ya que no depende de c y solo queremos el máximo.

$$\sum_{p\in\{b,a\}} P(C = c, F = f, X = d, D = s, P = p) =$$

$$\sum_{p\in\{b,a\}} P(P = p) \cdot P(F = f) \cdot P(C = c|P = p, F = f) \cdot P(X = d|C = c)P(D = s|C = c) =$$

El ejercicio está resuelto en el pdf del examen.