Aplicación de Modelos Ocultos de Markov (HMM) al reconocimiento del alfabeto latino

Tabla de contenidos

1 Reconocimiento de Caracteres

2 Espacio Probabilístico y Procesos Estocásticos Subyacentes

Para establecer formalmente una Cadena de Markov a Tiempo Continuo (CTMC), es necesario partir de los axiomas fundamentales de la teoría de la medida, para más detalle consultar el libro de Rudin (1987).

Para esto, se asumirá que el lector tiene conocimientos previos sobre teoría de la medida como lo son las σ-álgebras, espacio muestral y medida de probabilidad, entre otros.

Definición (Espacio de Probabilidad Filtrado)

Sea $(\Omega, \mathcal{F}, \mathbb{P})$ un espacio de probabilidad, donde Ω es el espacio muestral, $\mathcal{F}$ es una σ-álgebra sobre Ω, y $\mathbb{P}$ es una medida de probabilidad. Sea $\mathbb{F} = \{\mathcal{F}_t\}_{t\geq 0}$ una filtración, es decir, una familia no decreciente de sub-σ-álgebras de $\mathcal{F}$ tal que $\mathcal{F}_s \subseteq \mathcal{F}_t \subseteq \mathcal{F}$ para todo $0 \leq s < t$. El conjunto $(\Omega, \mathcal{F}, \mathbb{F}, \mathbb{P})$ se denomina espacio de probabilidad filtrado.

Definición (Espacio de Estados Discreto)

Definimos el espacio de estados S como un conjunto discreto y finito (o numerable). La topología en S es discreta y su σ-álgebra asociada es el conjunto potencia $\mathcal{P}(S)$.

En el contexto de nuestro proyecto, S representa el conjunto de posibles colores o intensidades de un pixel (por ejemplo, el espacio RGB donde $|S| = 256^3$, o una escala de grises finita).

Definición (Proceso Estocástico a Tiempo Continuo)

Un proceso estocástico en tiempo continuo con espacio de estados discreto S es una familia de variables aleatorias

$$X = \{X(t) : t \in \mathbb{R}_{\geq 0}\},$$

donde cada $X(t) : \Omega \to S$ es una función medible respecto a $\mathcal{F}_t$.

Interpretamos $X(t)$ como el "color verdadero" del pixel en el instante (o parámetro continuo) t.

3 Cadenas de Markov a Tiempo Continuo (CTMC)

La transición del concepto general de proceso estocástico al de proceso de Markov requiere la propiedad de amnesia (pérdida de memoria).

Definición (Propiedad de Markov)

El proceso adaptado $X = \{X(t)\}_{t\geq 0}$ es una Cadena de Markov a Tiempo Continuo si satisface la condición de Markov relativa a la filtración $\mathbb{F}$: para cualquier conjunto discreto de tiempos

$$0 \leq t_1 < t_2 < ... < t_n < t$$

y cualquier secuencia de estados

$$i_1, i_2, ..., i_n, i, j \in S,$$

se cumple que

$$\mathbb{P}(X(t) = j \mid X(t_1) = i_1, ..., X(t_n) = i, X(s) = i) = \mathbb{P}(X(t) = j \mid X(s) = i).$$

De manera equivalente y más elegante mediante la esperanza condicional:

$$\mathbb{P}(X(t) = j \mid \mathcal{F}_s) = \mathbb{P}(X(t) = j \mid X(s)) \quad \text{c.s. (casi seguramente)}$$

para todo $0 \leq s \leq t$.

Definición (Probabilidades de Transición y Homogeneidad)

Definimos la probabilidad de transición del estado i al estado j en el intervalo $[s, t]$ como

$$p_{ij}(s, t) = \mathbb{P}(X(t) = j \mid X(s) = i).$$

Se dice que la CTMC es **homogénea en el tiempo** si $p_{ij}(s, t)$ depende únicamente de la longitud del intervalo temporal

$$\tau = t - s.$$

En tal caso, denotamos la probabilidad de transición como

$$p_{ij}(\tau) = \mathbb{P}(X(s+\tau) = j \mid X(s) = i).$$

Sean estas probabilidades las entradas de una matriz estocástica

$$P(\tau) = [p_{ij}(\tau)]_{i,j \in S}.$$

! Propiedad (Ecuaciones de Chapman-Kolmogorov)

Para cualquier $s, t \geq 0$, la familia de matrices de transición $\{P(t)\}_{t \geq 0}$ forma un semigrupo de operadores estocásticos, satisfaciendo

$$P(t+s) = P(t)P(s)$$

lo que equivale a

$$p_{ij}(t+s) = \sum_{k \in S} p_{ik}(t) p_{kj}(s).$$

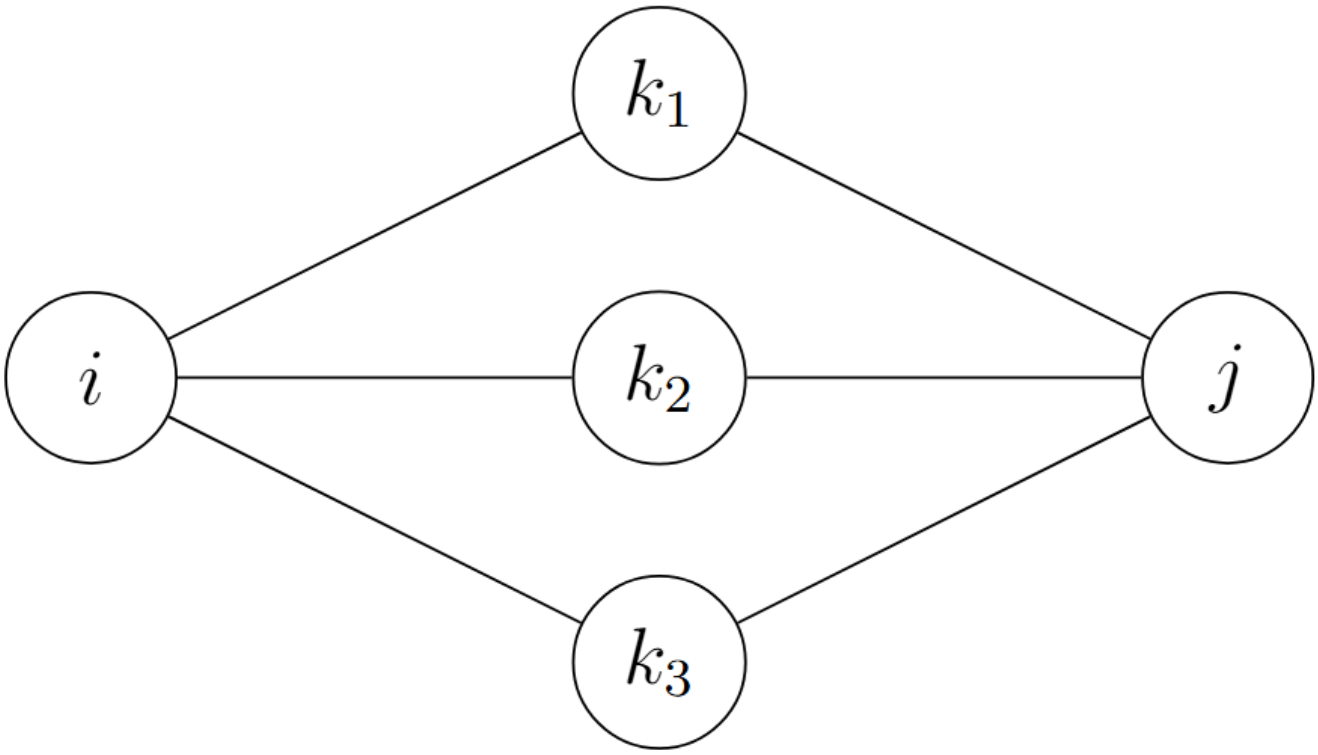


Figura 3.1: La probabilidad de transición $p\_ij(t+s)$ se obtiene sumando las contribuciones de todos los posibles estados intermedios k.

4 Dinámica Infinitesimal y Generador de la Cadena

La evolución continua del color del pixel requiere una descripción mediante probabilidades de transición, lo cual nos lleva a la representación matricial de la derivada de $P(t)$.

Definición (Matriz Generadora Infinitesimal)

Asumiendo que $p_{ij}(t)$ es diferenciable por la derecha en $t = 0$, definimos la tasa de transición q_{ij} del estado i al estado j (donde $i \neq j$) como:

$$q_{ij} = \lim_{h \to 0^+} \frac{p_{ij}(h)}{h} \geq 0.$$

Y para el caso $i = j$, definimos la probabilidad de salida del estado i como $v_i = -q_{ii}$, dada por

$$q_{ii} = \lim_{h \to 0^+} \frac{p_{ii}(h) - 1}{h} = -\sum_{j \neq i} q_{ij} \leq 0.$$

La matriz

$$Q = [q_{ij}]_{i,j \in S}$$

se denomina matriz generadora (o matriz Q) de la cadena. Sus filas suman cero:

$$\sum_{j \in S} q_{ij} = 0.$$

Propiedad (Ecuaciones Diferenciales de Kolmogorov)

Bajo condiciones de regularidad (que se cumplen trivialmente si el espacio de estados del color S es finito), la matriz de transición $P(t)$ es la solución única a las ecuaciones diferenciales matriciales de Kolmogorov Hacia Adelante (Forward) y Hacia Atrás (Backward) Norris (1998) :

- **Forward:**

$$\frac{d}{dt} P(t) = P(t)Q$$

- **Backward:**

$$\frac{d}{dt}P(t) = QP(t)$$

Cuya solución formal, bajo la condición inicial $P(0) = I$ (matriz identidad), es la matriz exponencial

$$P(t) = e^{Qt} = \sum_{n=0}^{\infty} \frac{(Qt)^n}{n!}.$$

5 Utilidad hacia Cadenas de Márkov Ocultas (HMM)

La fundamentación teórica precedente modela la dinámica subyacente determinista de la matriz de transición y la evolución estocástica de las trayectorias del sistema. Para extender esta arquitectura a un Modelo Oculto de Márkov (HMM) en el contexto de la inferencia y análisis de píxeles en imágenes que contienen letras del alfabeto latino, se formaliza la siguiente estructura bivariada:

Definición (Proceso Latente e Inobservable)

El proceso de Markov a tiempo continuo

$$X = \{X(t)\}_{t\geq 0}$$

sobre el espacio de estados S, caracterizado por su matriz generadora infinitesimal Q, constituye el *proceso latente*. Las alteraciones estocásticas inducidas por Q a lo largo del parámetro t (representando una métrica de degradación física o divergencia espacial) rigen las transiciones hacia estados degradados, manteniéndose inaccesibles a la medición directa.

La variable $X(t)$ describe el estado ideal o "verdadero" del pixel.

Definición (Proceso de Observación Empírica)

Sea

$$(O, \mathcal{P}(O))$$

el espacio medible de las posibles mediciones obtenidas por instrumentos.
Se postula un *proceso de observación* acoplado

$$Y = \{Y(t)\}_{t\geq 0},$$

compuesto por variables aleatorias

$$Y(t) : \Omega \to O$$

que representan la lectura empírica o ruidosa en el parámetro t.

En este caso O denota el conjunto finito de colores o intensidades registrables por un sensor óptico y Y es la lectura ruidosa del pixel en el tiempo t.

Definición (Matriz de Emisión e Independencia Condicional)

La dependencia estocástica entre ambos procesos queda unívocamente determinada por la condición de independencia local.
Para cualquier instante $t \geq 0$, la observación $Y(t)$ es condicionalmente independiente de toda la historia previa del sistema dado el estado concurrente $X(t)$.
Formalmente, para cualquier $y \in O$ y $x \in S$:

$$\mathbb{P}(Y(t) = y \mid \mathcal{F}_t^X, \mathcal{F}_{t^-}^Y) = \mathbb{P}(Y(t) = y \mid X(t) = x) \quad \text{c.s.}$$

donde $\mathcal{F}_t^X$ y $\mathcal{F}_{t^-}^Y$ denotan las filtraciones del historial latente y observable, respectivamente.
Estas probabilidades definen las entradas de la matriz de emisión estocástica

$$E = [e_{xy}],$$

donde

$$e_{xy} = \mathbb{P}(Y(t) = y \mid X(t) = x).$$

6 Inferencia y Filtrado en el Modelo Oculto

Una vez establecida la dinámica del sistema, es necesario desarrollar la maquinaria analítica para inferir el estado latente del pixel a partir de las observaciones empíricas.

Supongamos que disponemos de una secuencia de observaciones

$$Y = (y_1, y_2, \dots, y_K)$$

en instantes discretos de tiempo (o espacio)

$$0 \leq t_1 < t_2 < \dots < t_K \leq T.$$

Definición (Filtro Hacia Adelante (Forward) y Verosimilitud)

Definimos la variable de avance $\alpha_k(i)$ como la probabilidad conjunta de observar la secuencia parcial hasta el k-ésimo instante y que el estado oculto en dicho instante sea $i \in S$:

$$\alpha_k(i) = \mathbb{P}(Y(t_1) = y_1, \dots, Y(t_k) = y_k, X(t_k) = i).$$

Esta cantidad se calcula recursivamente mediante la ecuación de Chapman-Kolmogorov y la matriz de emisión:

$$\alpha_k(j) = \left(\sum_{i \in S} \alpha_{k-1}(i)\, p_{ij}(\Delta t_k) \right) \mathbb{P}(Y(t_k) = y_k \mid X(t_k) = j)$$

donde

$$\Delta t_k = t_k - t_{k-1}$$

y

$$p_{ij}(\Delta t_k)$$

es la entrada correspondiente de la matriz de transición

$$P(\Delta t_k) = e^{Q \Delta t_k}.$$

La verosimilitud total de la secuencia de observaciones es simplemente

$$\mathbb{P}(Y) = \sum_{i \in S} \alpha_K(i).$$

Definición (Decodificación Óptima Global)

Para reconstruir la trayectoria más probable, buscamos la secuencia de estados latentes

$$\hat{X} = (\hat{x}_1, \dots, \hat{x}_K)$$

que maximice la probabilidad condicional conjunta

$$\hat{X} = \arg \max_{x_1,\dots,x_K} \mathbb{P}\Big(X(t_1) = x_1, \dots, X(t_K) = x_K \mid Y(t_1) = y_1, \dots, Y(t_K) = y_K\Big).$$

La solución sistemática a este problema de programación dinámica espacial/temporal se obtiene mediante la adaptación del Algoritmo de Viterbi para cadenas a tiempo continuo, evaluando las transiciones máximas sobre

$$P(\Delta t_k).$$

Para los pixeles es la trayectoria más probable de los colores verdaderos del pixel (el problema de decodificación).

6.1. Ejemplo ilustrativo del algoritmo Forward–Backward

Para ilustrar el funcionamiento de los algoritmos Forward y Backward en el contexto del reconocimiento de imágenes, consideremos un píxel cuyo color verdadero puede encontrarse en uno de dos estados ocultos:

$$S = \{B, N\},$$

donde:

- B: píxel verdaderamente blanco.
- N: píxel verdaderamente negro.

Debido al ruido en el proceso de adquisición de la imagen, el color observado puede diferir del color real.

Supongamos que la dinámica del píxel está descrita por la matriz de transición

$$P = \begin{pmatrix} 0.8 & 0.2 \\ 0.3 & 0.7 \end{pmatrix},$$

donde, por ejemplo,

$$p_{BN} = 0.2$$

representa la probabilidad de que un píxel blanco pase al estado negro en el siguiente instante.

Asimismo, consideremos la siguiente matriz de emisión:

$$E = \begin{pmatrix} 0.9 & 0.1 \\ 0.2 & 0.8 \end{pmatrix},$$

cuyas columnas corresponden a las observaciones

$$(O_B, O_N),$$

es decir, observar blanco u observar negro. Por ejemplo,

$$P(O_B \mid B) = 0.9, \qquad P(O_B \mid N) = 0.2.$$

Finalmente, asumimos una distribución inicial

$$\pi = (0.6, 0.4),$$

y que la secuencia observada es

$$Y = (O_B, O_N).$$

Esto significa que el sensor registra blanco en el primer instante y negro en el segundo.

6.1.1. Cálculo Forward

La variable Forward se define como

$$\alpha_k(i) = P(y_1, \dots, y_k, X_k = i).$$

6.1.1.1. Paso 1

Para la primera observación O_B:

$$\alpha_1(B) = P(X_1 = B)\, P(O_B \mid B) = 0.6(0.9) = 0.54,$$

$$\alpha_1(N) = P(X_1 = N)\, P(O_B \mid N) = 0.4(0.2) = 0.08.$$

6.1.1.2. Paso 2

Calculamos primero la probabilidad de alcanzar el estado blanco:

$$\sum_{i \in S} \alpha_1(i) p_{iB} = 0.54(0.8) + 0.08(0.3) = 0.456.$$

Multiplicando por la probabilidad de emisión correspondiente,

$$\alpha_2(B) = 0.456(0.1) = 0.0456.$$

Análogamente, para el estado negro:

$$\sum_{i \in S} \alpha_1(i) p_{iN} = 0.54(0.2) + 0.08(0.7) = 0.164,$$

y por tanto,

$$\alpha_2(N) = 0.164(0.8) = 0.1312.$$

6.1.1.3. Verosimilitud de la secuencia observada

La probabilidad total de observar la secuencia Y es

$$P(Y) = \alpha_2(B) + \alpha_2(N),$$

es decir,

$$P(Y) = 0.0456 + 0.1312 = 0.1768.$$

6.1.2. Cálculo Backward

La variable Backward se define mediante

$$\beta_k(i) = P(y_{k+1}, \ldots, y_T \mid X_k = i).$$

Como la secuencia contiene únicamente dos observaciones,

$$\beta_2(B) = \beta_2(N) = 1.$$

Retrocediendo un paso:

$$\beta_1(B) = 0.8(0.1)(1) + 0.2(0.8)(1) = 0.24,$$

y

$$\beta_1(N) = 0.3(0.1)(1) + 0.7(0.8)(1) = 0.59.$$

6.1.3. Inferencia del estado oculto

Una de las principales ventajas del algoritmo Forward–Backward es que permite calcular la probabilidad posterior de cada estado oculto.

Para el estado blanco en el primer instante,

$$\gamma_1(B) = P(X_1 = B \mid Y) = \frac{\alpha_1(B)\beta_1(B)}{P(Y)},$$

por lo que

$$\gamma_1(B) = \frac{0.54(0.24)}{0.1768} = 0.733.$$

De forma análoga,

$$\gamma_1(N) = \frac{\alpha_1(N)\beta_1(N)}{P(Y)} = \frac{0.08(0.59)}{0.1768} = 0.267.$$

En consecuencia,

$$P(X_1 = B \mid Y) \approx 73.3\,\%,$$

mientras que

$$P(X_1 = N \mid Y) \approx 26.7\,\%.$$

Interpretación

A pesar de que el sensor observa un píxel blanco en el primer instante y negro en el segundo, el algoritmo Forward–Backward determina que existe aproximadamente un 73 % de probabilidad de que el color verdadero inicial haya sido blanco. Este resultado muestra cómo un Modelo Oculto de Markov combina la información proporcionada por las observaciones ruidosas con la dinámica de transición del sistema para inferir el estado real subyacente del píxel.

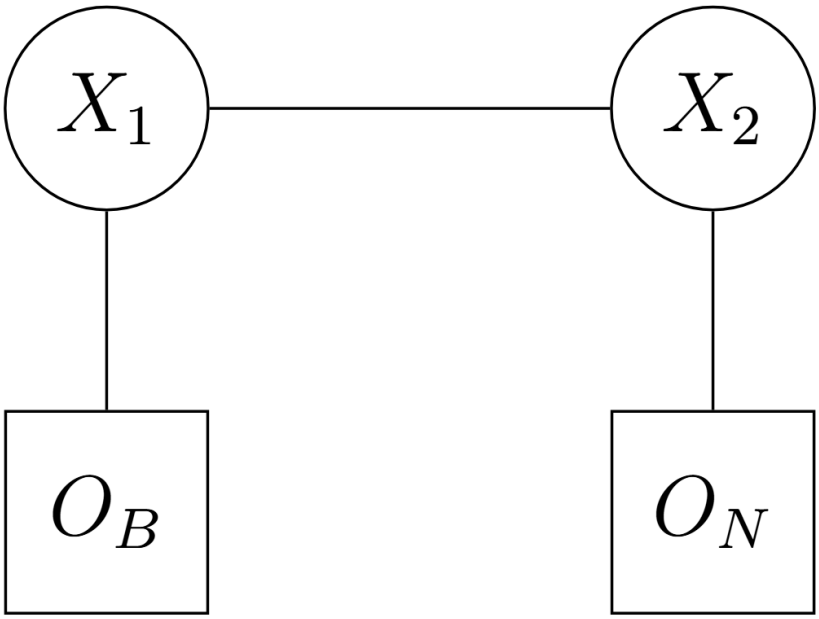


Figura 6.1: Modelo Oculto de Markov para un píxel con dos estados ocultos y dos observaciones.

7 Estimación de Parámetros y Calibración del Modelo

En la práctica, la matriz generadora infinitesimal Q (que dicta la tasa de degradación del color para el caso de estudio) y las probabilidades de emisión empírica rara vez son conocidas a priori y deben ser estimadas a partir de un dataset de imágenes (el alfabeto latino modificado).

! Propiedad (Algoritmo de Expectation-Maximization (Baum-Welch))

La estimación por máxima verosimilitud del conjunto de parámetros del modelo

$$\Theta = \{Q, E\}$$

dado un conjunto de secuencias observadas se realiza mediante un proceso iterativo.

- **Paso E (Expectation):** Se calculan las estadísticas suficientes esperadas del sistema latente (tiempo esperado de permanencia en cada estado de color y número esperado de transiciones) condicionado a las observaciones y a la estimación actual de los parámetros $\Theta^{(m)}$, utilizando las variables recursivas Forward y Backward.
- **Paso M (Maximization):** Se actualizan los parámetros $\Theta^{(m+1)}$ para maximizar la función Q de verosimilitud esperada. En particular, la estimación de las probabilidades de transición empíricas se formula restringiendo las soluciones para garantizar que las filas de $Q^{(m+1)}$ sigan sumando cero y $q_{ij} \geq 0$ para $i \neq j$.

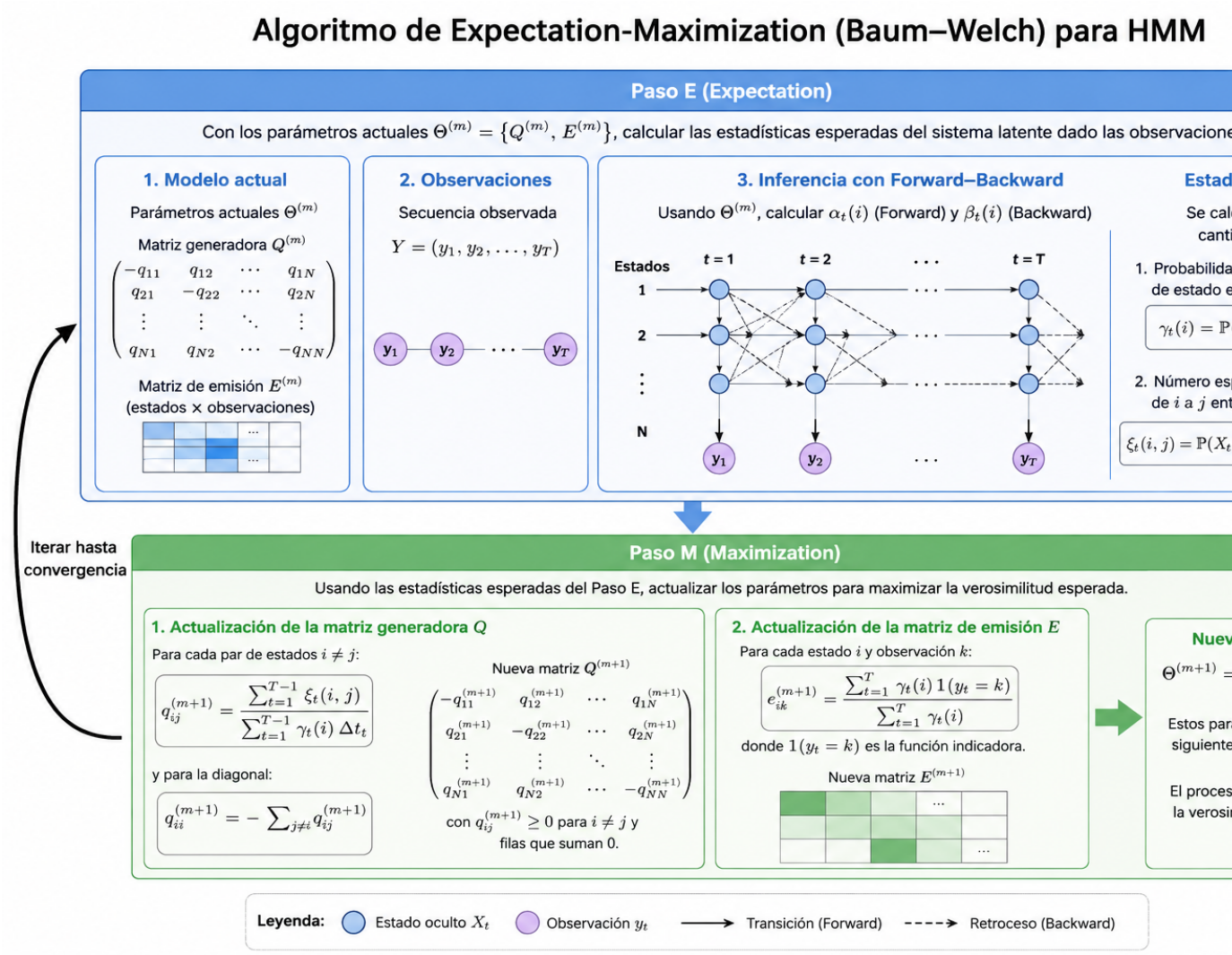


Figura 7.1: Esquema general del algoritmo Expectation–Maximization (Baum–Welch) para un Modelo Oculto de Markov. En el Paso E se estiman las probabilidades posteriores de los estados ocultos y las transiciones esperadas mediante los algoritmos Forward–Backward. En el Paso M se actualizan iterativamente la matriz generadora y la matriz de emisión para maximizar la verosimilitud de las observaciones.

8 Caso de estudio: Reconocimiento de Caracteres Manuscritos con HMM

9 Reconocimiento de Caracteres Caso de estudio

En este capítulo se describe la metodología implementada para el reconocimiento automático de letras mayúsculas manuscritas del alfabeto latino mediante Modelos Ocultos de Markov (HMM). A diferencia del capítulo anterior, donde se presentaron los fundamentos teóricos del modelo, en esta sección se detallan las decisiones experimentales adoptadas, así como las etapas seguidas para la construcción, entrenamiento y evaluación del sistema de reconocimiento.

Con el propósito de garantizar la reproducibilidad del experimento, se presenta el código utilizado durante el desarrollo del proyecto. Debido al elevado tiempo de ejecución requerido para el entrenamiento de los modelos, los bloques de código se muestran únicamente con fines documentales y no son ejecutados automáticamente dentro del presente documento.

9.1. Configuración del entorno experimental

El primer paso consistió en importar las bibliotecas necesarias para el procesamiento de imágenes, manipulación de datos, entrenamiento de los modelos HMM y evaluación de resultados. Asimismo, se estableció una conexión con un sistema de almacenamiento externo para conservar los modelos entrenados y evitar repetir el proceso de ajuste en ejecuciones posteriores.

```
from google.colab import drive
drive.mount('/content/drive')
import cv2
import numpy as np
import torchvision.datasets as datasets
import torchvision.transforms as transforms
import matplotlib.pyplot as plt
import seaborn as sns
import random
import os
import pickle
from tqdm import tqdm
try:
    from hmmlearn import hmm
except ImportError:
```

```
    import subprocess
    import sys
    subprocess.check_call([sys.executable, "-m", "pip", "install", "hmmlearn"])
    from hmmlearn import hmm
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
import warnings
```

La utilización de estas bibliotecas responde a necesidades específicas del sistema. `OpenCV` permitió realizar operaciones de preprocesamiento sobre las imágenes; `torchvision` proporcionó acceso al conjunto de datos **EMNIST** (Cohen et al. 2017); `hmmlearn` hizo posible la implementación de los modelos ocultos de Markov; mientras que `scikit-learn` facilitó el cálculo de las métricas de evaluación.

Con el fin de asegurar la reproducibilidad de los resultados, se fijaron semillas aleatorias y se deshabilitaron advertencias no críticas del entorno de ejecución.

```
np.random.seed(42)
random.seed(42)
warnings.filterwarnings("ignore")
```

El establecimiento de semillas garantiza que los procesos dependientes del azar produzcan resultados consistentes entre diferentes ejecuciones, permitiendo replicar los experimentos bajo las mismas condiciones iniciales.

9.2. Definición de parámetros del experimento

A continuación, se definieron los parámetros principales utilizados durante el entrenamiento y evaluación del sistema.

```
N_ESTADOS = 12
TRAIN_POR_CLASE = 1500
TEST_POR_CLASE = 400

IMG_SIZE = 40
UMBRAL = 0.30

ARCHIVO_MODELOS_CAT = "/content/drive/MyDrive/modelos_categorical_v_avanzada_hmm.pkl"

TARGETS_MAYUSCULAS = list(range(10,36))
MAPEO_LETTERS = {i: chr(ord('A') + (i - 10)) for i in range(10,36)}
LETRAS_OBJETIVO = [chr(ord('A') + i) for i in range(26)]
```

Se optó por utilizar 12 estados ocultos para representar la evolución secuencial de cada letra manuscrita. Este número proporciona un equilibrio entre la capacidad descriptiva del modelo y el costo computacional asociado al entrenamiento. Asimismo, se seleccionaron 1500 muestras de entrenamiento y 400 muestras de prueba por clase con el objetivo de mantener una representación equilibrada de todas las letras del alfabeto.

El tamaño de imagen fue fijado en 40×40 píxeles, permitiendo estandarizar las observaciones y reducir la variabilidad asociada a las dimensiones originales de escritura.

9.3. Obtención del conjunto de datos

Para el entrenamiento y evaluación del sistema se empleó el conjunto de datos **EMNIST ByClass**, una extensión del conocido conjunto **MNIST** que incorpora letras manuscritas y dígitos.

```
print("Descargando EMNIST ByClass...")
emnist_train = datasets.EMNIST(root='./data', split='byclass', train=True, download=True,
emnist_test = datasets.EMNIST(root='./data', split='byclass', train=False, download=True,
```

A partir de este conjunto se seleccionaron únicamente las letras mayúsculas, descartando las demás clases disponibles. Esta decisión se tomó con el propósito de acotar el problema de clasificación y evaluar específicamente la capacidad del modelo para reconocer caracteres alfabéticos manuscritos.

9.4. Definición de la arquitectura del HMM

Se construyó una matriz de transición tipo Bakis (falta citar) flexible para modelar la evolución progresiva de los trazos durante la generación de cada letra.

```
def construir_matriz_bakis_flexible(n_states):
    A = np.zeros((n_states, n_states))
    for i in range(n_states):
        if i == n_states - 1:
            A[i, i] = 1.0
        elif i == n_states - 2:
            A[i, i] = 0.60
            A[i, i+1] = 0.40
        else:
            A[i, i] = 0.60
            A[i, i+1] = 0.30
            A[i, i+2] = 0.10
    return A
```

La arquitectura Bakis impone una restricción direccional sobre las transiciones, permitiendo permanecer en el mismo estado o avanzar hacia estados posteriores, pero evitando retrocesos. Esta estructura resulta adecuada para describir procesos con una evolución secuencial, como la formación de los trazos manuscritos.

9.5. Preprocesamiento de imágenes

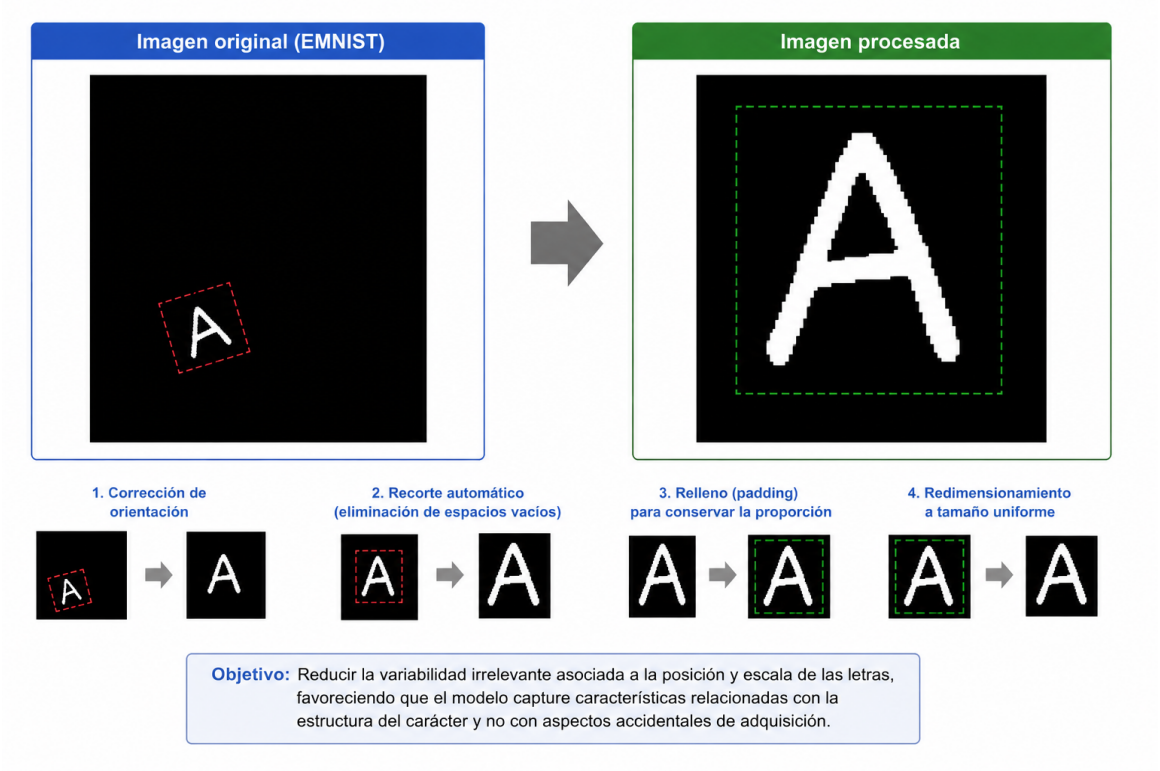
Antes de convertir las imágenes en secuencias observables, se aplicó un proceso de normalización y alineamiento.

```
def procesar_imagen(imagen_tensor):
    imagen = imagen_tensor.squeeze().numpy()
    imagen = np.fliplr(imagen)
    imagen = np.rot90(imagen)

    img_bin = (imagen > UMBRAL).astype(np.uint8)
    filas = np.any(img_bin, axis=1)
    columnas = np.any(img_bin, axis=0)

    if np.any(filas) and np.any(columnas):
        ymin, ymax = np.where(filas)[0][[0,-1]]
        xmin, xmax = np.where(columnas)[0][[0,-1]]
        recorte = imagen[ymin:ymax+1, xmin:xmax+1]

        h, w = recorte.shape
        max_dim = max(h, w)
        pad_h = max_dim - h
        pad_w = max_dim - w

        pad_top = pad_h // 2
        pad_bottom = pad_h - pad_top
        pad_left = pad_w // 2
        pad_right = pad_w - pad_left

        recorte_cuadrado = np.pad(recorte, ((pad_top, pad_bottom), (pad_left, pad_right)),
        imagen = cv2.resize(recorte_cuadrado.astype(np.float32), (IMG_SIZE, IMG_SIZE), int
    else:
        imagen = cv2.resize(imagen.astype(np.float32), (IMG_SIZE, IMG_SIZE), interpolation

    imagen = (imagen > UMBRAL).astype(float)
    return imagen
```

El procedimiento incluyó la corrección de orientación propia del conjunto **EMNIST**, la eliminación de espacios vacíos mediante recortes automáticos, la incorporación de relleno

para conservar la proporción del carácter y el redimensionamiento a un tamaño uniforme.

Estas transformaciones buscan reducir la variabilidad irrelevante asociada a la posición y escala de las letras, favoreciendo que el modelo capture características relacionadas con la estructura del carácter y no con aspectos accidentales de adquisición.



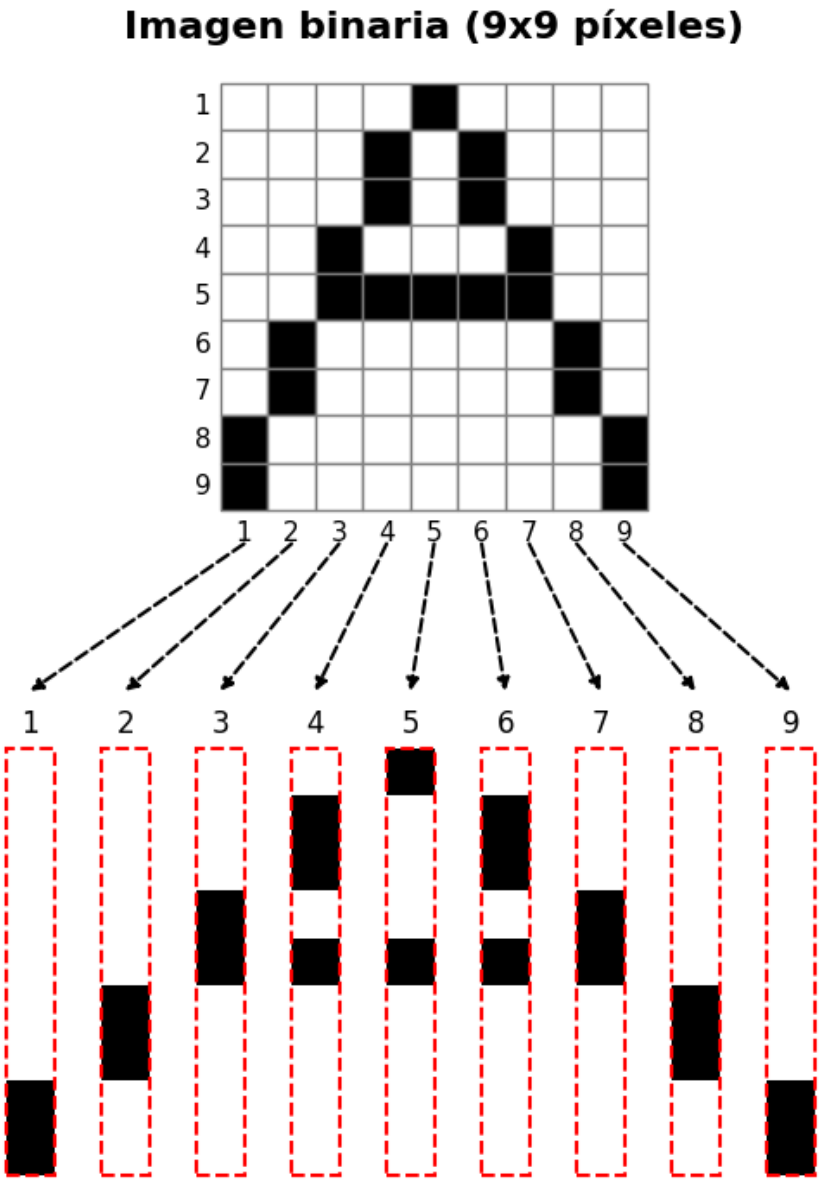
9.6. Extracción de características

Posteriormente, cada imagen preprocesada fue transformada en una secuencia discreta de observaciones.

```
def imagen_a_secuencia(img, w=5, stride=2):
    secuencia = []
    altura = IMG_SIZE

    for t in range(0, img.shape[1] - w + 1, stride):
        ventana = img[:, t:t+w]
        columna = np.mean(ventana, axis=1)
        col_bin = (columna > 0.35).astype(int)

        suma_tinta = np.sum(col_bin)
```

```
        if suma_tinta == 0:
            simbolo = 0
            secuencia.append([simbolo])
            continue

        if suma_tinta < 8: tinta_q = 0
        elif suma_tinta < 16: tinta_q = 1
        elif suma_tinta < 24: tinta_q = 2
        else: tinta_q = 3

        cg = np.sum(col_bin * np.arange(altura)) / suma_tinta
        if cg < 12: cg_q = 0
        elif cg < 20: cg_q = 1
        elif cg < 28: cg_q = 2
        else: cg_q = 3

        transiciones = np.sum(np.abs(np.diff(col_bin)))
        trans_q = min(int(transiciones), 3)

        indices_tinta = np.where(col_bin > 0)[0]
        perfil_sup = indices_tinta[0]
        perfil_inf = indices_tinta[-1]

        up_q = 1 if perfil_sup > 10 else 0
        low_q = 1 if perfil_inf < 30 else 0

        simbolo = (tinta_q << 6) | (cg_q << 4) | (trans_q << 2) | (up_q << 1) | low_q
        secuencia.append([simbolo])

    return np.array(secuencia)
```

La extracción se realizó mediante ventanas deslizantes verticales. Para cada ventana se calcularon características geométricas relacionadas con la distribución de "tinta", el centro de gravedad, el número de transiciones entre regiones activas e inactivas y la ubicación relativa de los perfiles superior e inferior del trazo.



Cada columna de píxeles constituye una ventana vertical utilizada para construir la secuencia de observaciones (Matriz de Bakis).

Dichas características fueron cuantificadas y codificadas en un conjunto discreto de 256 símbolos observables. Este procedimiento permitió adaptar el problema a la formulación categórica de HMM, cuyos mecanismos de emisión operan sobre observaciones discretas.

9.7. Construcción de los conjuntos de entrenamiento y prueba

Las secuencias obtenidas fueron organizadas en conjuntos balanceados de entrenamiento y evaluación.

```
def construir_dataset(dataset, max_muestras):
    X, y = [], []
    conteos = {k:0 for k in TARGETS_MAYUSCULAS}

    for tensor, target in dataset:
        if target not in TARGETS_MAYUSCULAS: continue
        if conteos[target] >= max_muestras: continue

        img = procesar_imagen(tensor)
        seq = imagen_a_secuencia(img)

        X.append(seq)
        y.append(MAPEO_LETTERS[target])
        conteos[target] += 1

        if all(c >= max_muestras for c in conteos.values()): break

    return X, np.array(y)

print("Construyendo entrenamiento...")
X_train, y_train = construir_dataset(emnist_train, TRAIN_POR_CLASE)

print("Construyendo prueba...")
X_test, y_test = construir_dataset(emnist_test, TEST_POR_CLASE)
```

El balance entre clases evita sesgos hacia letras con mayor número de muestras disponibles y permite interpretar las métricas de desempeño de manera más objetiva.

9.8. Entrenamiento de los modelos

Se entrenó un modelo HMM independiente para cada una de las letras del alfabeto.

```
modelos_hmm = {}

if os.path.exists(ARCHIVO_MODELOS_CAT):
    print("\n[INFO] Modelos detectados en Drive. Cargando...")
    with open(ARCHIVO_MODELOS_CAT, 'rb') as f:
        modelos_hmm = pickle.load(f)
```

```
else:
    print("\nEntrenando CategoricalHMMs...\n")
    barra = tqdm(LETRAS_OBJETIVO, desc="Entrenando", unit="letra")
    for letra in barra:
        barra.set_description(f"Entrenando '{letra}'")
        idx = np.where(y_train == letra)[0]
        secuencias = [X_train[i] for i in idx]
        lengths = [len(s) for s in secuencias]
        X_concat = np.vstack(secuencias)

        model = hmm.CategoricalHMM(
            n_components=N_ESTADOS,
            n_features=256,
            n_iter=250,
            random_state=42,
            init_params='e'
        )

        model.startprob_ = np.zeros(N_ESTADOS)
        model.startprob_[0] = 1.0
        model.transmat_ = construir_matriz_bakis_flexible(N_ESTADOS)

        model.fit(X_concat, lengths)
        modelos_hmm[letra] = model

    with open(ARCHIVO_MODELOS_CAT, 'wb') as f:
        pickle.dump(modelos_hmm, f)
```

Cada modelo fue ajustado utilizando exclusivamente las secuencias correspondientes a su letra asociada. Esta estrategia de clasificación por modelos independientes permite estimar la probabilidad de que una secuencia haya sido generada por cada carácter y seleccionar posteriormente la hipótesis más probable.

Los modelos entrenados fueron almacenados en un sistema externo de persistencia, evitando repetir el costoso proceso de entrenamiento en futuras ejecuciones.

9.9. Inferencia y clasificación

Una vez entrenados los modelos, cada muestra del conjunto de prueba fue evaluada frente a todos los HMM disponibles.

```
print("\nClasificando...\n")
y_pred = []
```

```
for seq in tqdm(X_test, desc="Evaluando Inferencia"):
    scores = {}
    for letra, model in modelos_hmm.items():
        try: scores[letra] = model.score(seq)
        except: scores[letra] = -np.inf

    pred = max(scores, key=scores.get)
    y_pred.append(pred)
```

La predicción final se obtuvo seleccionando la letra cuyo modelo produjo la mayor puntuación de verosimilitud sobre la secuencia observada.

Este criterio corresponde al principio de máxima verosimilitud y constituye una de las estrategias más utilizadas en tareas de reconocimiento basadas en HMM.

9.10. Evaluación del desempeño

Finalmente, se calcularon diferentes métricas para evaluar el rendimiento del sistema.

```
accuracy = accuracy_score(y_test, y_pred)
cm = confusion_matrix(y_test, y_pred, labels=LETRAS_OBJETIVO)

precision_por_letra = cm.diagonal() / cm.sum(axis=1)

print("\n" + "="*70)
print("EXACTITUD (ACCURACY) INDIVIDUAL POR LETRA")
print("="*70)
for i, letra in enumerate(LETRAS_OBJETIVO):
    acc = precision_por_letra[i] * 100
    print(f"Precisión Letra {letra} : {acc:>6.2f} %")

print("\n" + "="*70)
print(f"ACCURACY GLOBAL: {accuracy*100:.2f}%")
print("="*70 + "\n")

print(classification_report(y_test, y_pred, target_names=LETRAS_OBJETIVO))
```

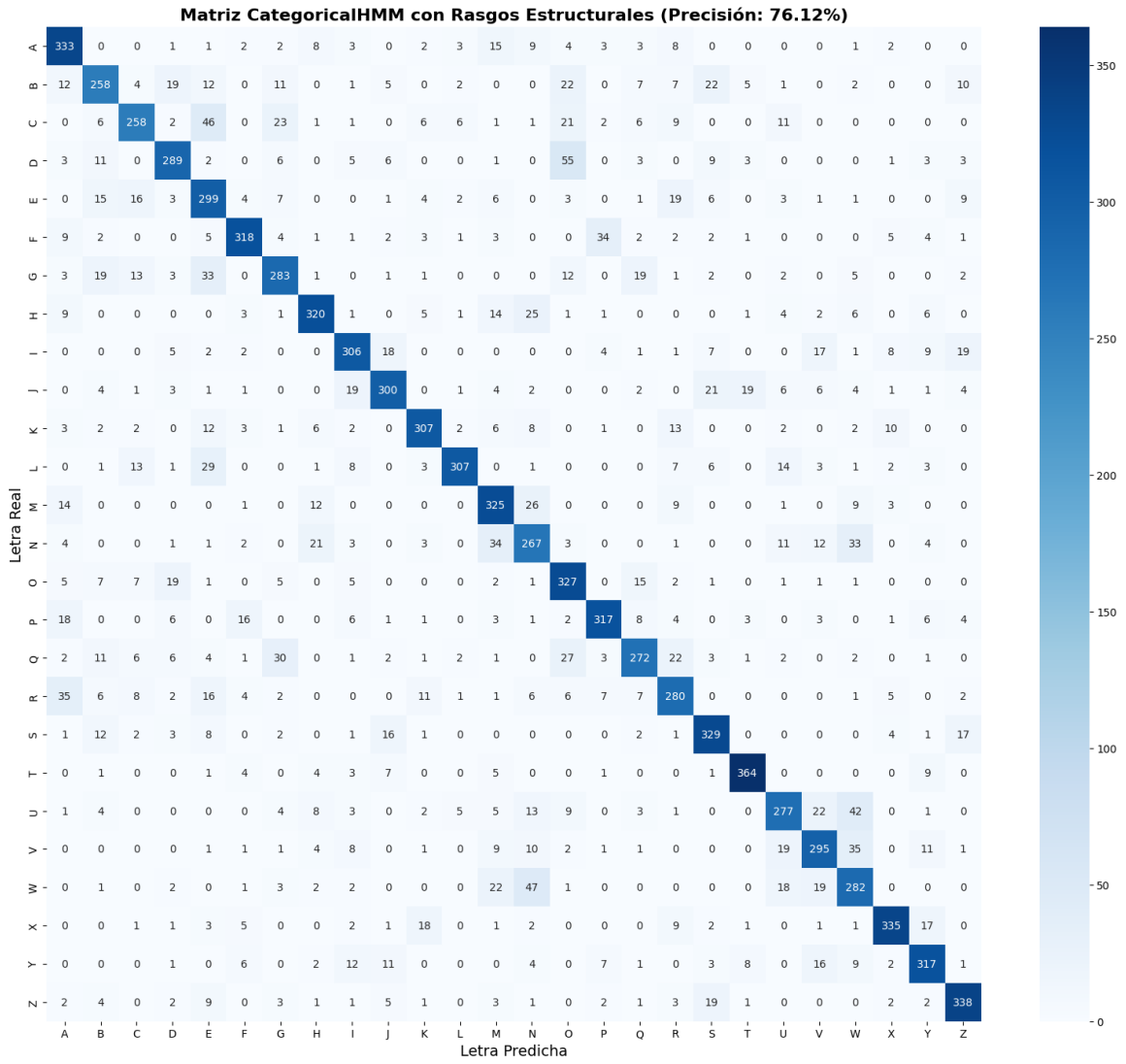
```
======================================================================
EXACTITUD (ACCURACY) INDIVIDUAL POR LETRA
======================================================================
Precisión Letra A :  83.25 %
Precisión Letra B :  64.50 %
Precisión Letra C :  64.50 %
Precisión Letra D :  72.25 %
Precisión Letra E :  74.75 %
Precisión Letra F :  79.50 %
Precisión Letra G :  70.75 %
Precisión Letra H :  80.00 %
Precisión Letra I :  76.50 %
Precisión Letra J :  75.00 %
Precisión Letra K :  80.37 %
Precisión Letra L :  76.75 %
Precisión Letra M :  81.25 %
Precisión Letra N :  66.75 %
Precisión Letra O :  81.75 %
Precisión Letra P :  79.25 %
Precisión Letra Q :  68.00 %
Precisión Letra R :  70.00 %
Precisión Letra S :  82.25 %
Precisión Letra T :  91.00 %
Precisión Letra U :  69.25 %
Precisión Letra V :  73.75 %
Precisión Letra W :  70.50 %
Precisión Letra X :  83.75 %
Precisión Letra Y :  79.25 %
Precisión Letra Z :  84.50 %
```

```
======================================================================
ACCURACY GLOBAL: 76.12%
======================================================================
```

	precision	recall	f1-score	support
A	0.73	0.83	0.78	400
B	0.71	0.65	0.68	400
C	0.78	0.65	0.71	400
D	0.78	0.72	0.75	400
E	0.62	0.75	0.67	400
F	0.85	0.80	0.82	400
G	0.73	0.71	0.72	400
H	0.82	0.80	0.81	400
I	0.78	0.77	0.77	400
J	0.80	0.75	0.77	400
K	0.83	0.80	0.82	382
L	0.92	0.77	0.84	400
M	0.70	0.81	0.75	400
N	0.63	0.67	0.65	400
O	0.66	0.82	0.73	400
P	0.83	0.79	0.81	400
Q	0.77	0.68	0.72	400
R	0.70	0.70	0.70	400
S	0.76	0.82	0.79	400
T	0.89	0.91	0.90	400
U	0.74	0.69	0.72	400
V	0.74	0.74	0.74	400
W	0.64	0.70	0.67	400
X	0.88	0.84	0.86	400
Y	0.80	0.79	0.80	400
Z	0.82	0.84	0.83	400
accuracy			0.76	10382
macro avg	0.77	0.76	0.76	10382
weighted avg	0.77	0.76	0.76	10382

Además del porcentaje global de aciertos, se estimó la precisión individual para cada letra y se construyó una matriz de confusión.

```
plt.figure(figsize=(16,14))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=LETRAS_OBJETIVO, yticklabel
plt.title(f'Matriz CategoricalHMM con Rasgos Estructurales (Precisión: {accuracy*100:.2f}%
plt.ylabel('Letra Real', fontsize=14)
plt.xlabel('Letra Predicha', fontsize=14)
plt.tight_layout()
plt.show()
```



La matriz de confusión permitió identificar cuáles letras presentan mayores dificultades de discriminación, proporcionando información valiosa para analizar las fortalezas y limitaciones del sistema propuesto. En conjunto, estas métricas ofrecen una evaluación integral del desempeño alcanzado por el modelo de reconocimiento desarrollado.

Cohen, Gregory, Saeed Afshar, Jonathan Tapson, y André van Schaik. 2017. «EMNIST: An Extension of MNIST to Handwritten Letters». *arXiv preprint arXiv:1702.05373*. https://arxiv.org/abs/1702.05373.

Norris, J. R. 1998. *Markov Chains*. Cambridge: Cambridge University Press.

Rudin, Walter. 1987. *Real and Complex Analysis*. 3.ª ed. New York: McGraw-Hill.