Inteligencia Artificial 1

Universidad Nacional de Cuyo - Facultad de Ingeniería

Trabajo Práctico N°8

Trabajo Práctico N°8: Aprendizaje

Grupo 2: Avila J., Barrios F., Patricelli N.

Septiembre 2025

---

1 Temas Tratados en el Trabajo Práctico 8

- Aprendizaje estadístico.
- Evolución de la verosimilitud de una hipótesis en función de observaciones.
- Aprendizaje no supervisado. Algoritmo K-means.
- Aprendizaje supervisado. Algoritmo Knn.
- Aprendizaje por refuerzo. Algoritmo Q-Learning.

2 Ejercicios Teóricos

2.1 Un fabricante de tornillos vende cajas que contienen 1000 tornillos de cabeza redonda con tres tipos de recubrimiento electrolítico (cincado, cobre y níquel). Cada caja se rellena con diferentes proporciones que pueden variar de la siguiente manera:

a: Todos los tornillos están recubiertos de níquel. 15 de cada 100 cajas se llenan de esta manera.

b: El 70% de los tornillos están recubiertos de níquel, el 20% de cobre, el resto está cincado. 15 de cada 100 cajas se llenan de esta manera.

c: El 50% de los tornillos están recubiertos de níquel, el 25% de cobre y el resto está cincado. 50 de cada 100 cajas se llenan de esta manera.

d: El 20 % de los tornillos están recubiertos de níquel, el 50% de cobre y el resto está cincado. 10 de cada 100 cajas se llenan de esta manera.

e: Todos los tornillos están recubiertos de cobre. 10 de cada 100 cajas se llenan de esta manera.

2.1.1 ¿Cuál es la distribución a priori sobre las hipótesis?

La distribución a priori sobre las hipótesis son las frecuencias de tipos de caja, es decir, el dominio de la hipótesis son los tipos de caja: $\Omega_H = \{A, B, C, D, E\}$.

Luego, podemos plantear la siguiente tabla, teniendo en cuenta que la **frecuencia** es por **cada 100 cajas**:

Tipo de Caja	Frecuencia	Probabilidad a priori
A	15	15%
B	15	15%
C	50	50%
D	10	10%
E	10	10%

Esto lo interpretamos como: - $P(H = A) = 15\%$ - $P(H = B) = 15\%$ - $P(H = C) = 50\%$ - $P(H = D) = 10\%$ - $P(H = E) = 10\%$

2.1.2 Considerando que los 10 primeros tornillos que se extraen de una caja de muestra son de cobre: Calcule la probabilidad de cada hipótesis dado que los 10 primeros tornillos fueron de cobre.

Lo que queremos calcular es $P(H|E, E, ..., E)$ dado que sucedio $E = Cu$ 10 veces consecutivas.

Para ello, primero debemos calcular la verosimilitud $P(E = 10 \times Cu|H)$. Considerando que al sacar un tornillo de cobre, las probabilidades se modifican (sin reemplazo), planteamos:

$$P(Cu|A) = 0$$
$$P(10 \times Cu|A) = 0$$

$$P(Cu|B) = 0.2$$
$$P(10 \times Cu|B) = \prod_{j=0}^{9} \frac{200 - j}{1000 - j} = 8,5232 \times 10^{-8}$$

$$P(Cu|C) = 0.25$$
$$P(10 \times Cu|C) = \prod_{j=0}^{9} \frac{250 - j}{1000 - j} = 8,31425 \times 10^{-7}$$

$$P(Cu|D) = 0.5$$
$$P(10 \times Cu|D) = \prod_{j=0}^{9} \frac{500 - j}{1000 - j} = 9,33188 \times 10^{-4}$$

$$P(Cu|E) = 1$$
$$P(10 \times Cu|E) = 1$$

Luego, con ley de bayes podemos calcular las probabilidades de que ocurra H_i sabiendo que sucedió $E = Cu$ 10 veces consecutivas:

$$P(H_i|10 \times Cu) = \frac{P(H_i)P(10 \times Cu|H_i)}{\sum_j P(H_j)P(10 \times Cu|H_j)} = \frac{P(H_i)P(10 \times Cu|H_i)}{\alpha}$$

Primero, calculamos α :

$$\alpha = P(A)P(10 \times Cu|A) + P(B)P(10 \times Cu|B) + P(C)P(10 \times Cu|C) + P(D)P(10 \times Cu|D) + P(E)P(10 \times Cu|E)$$

$$\alpha \approx 0,1000937473$$

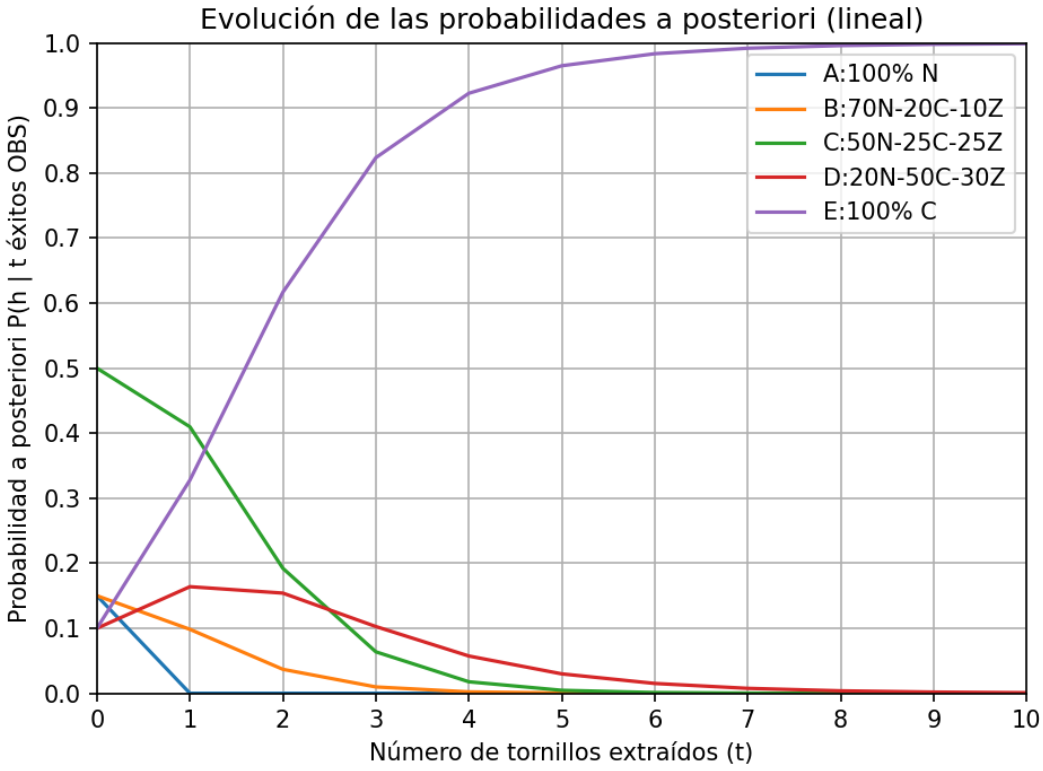
Y las probabilidades quedan:

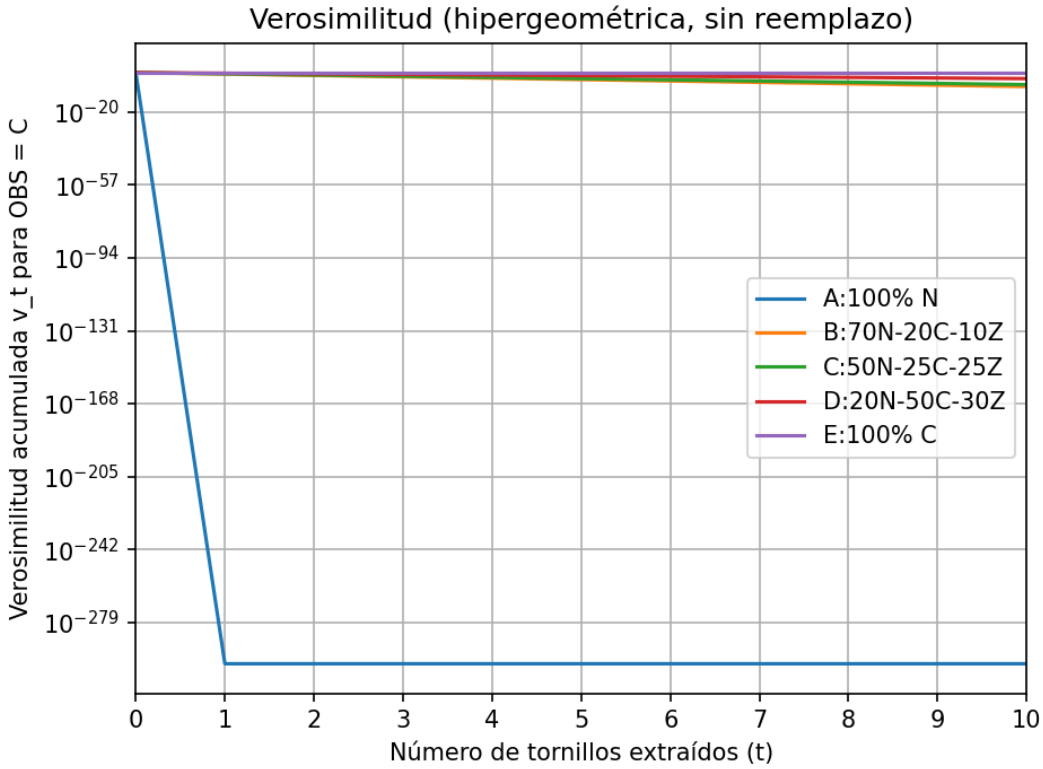
Caja	Probabilidad
A	0%
B	$1,277 \times 10^{-5}$ %
C	$4,153 \times 10^{-4}$ %
D	$9,323 \times 10^{-2}$ %
E	99,906 %

Usando el sentido común, es obvio que si solo saco tornillos consecutivos con recubrimiento de cobre, la opción más probable es que sea la caja que esta constituidas sola y exclusivamente de tornillos recubiertos de cobre. Curiosamente, toma solo un tornillo con otro recubrimiento para que la probabilidad se vuelva cero (ya que si hay 2 tipos de recubrimiento diferentes que salieron de una misma caja, obviamente no puede ser una caja que contiene exclusivamente tornillos de un único tipo de recubrimiento).

2.1.3 Grafique la evolución de la verosimilitud de cada hipótesis en función del número de tornillos extraídos de la caja.

Las gráficas son:





2.1.4 Para cada hipótesis, ¿cuál es la probabilidad de que el cuarto tornillo extraído sea de cobre?

Depende cómo se plantee la respuesta, podemos desarrollarlo de dos formas distintas:

Sin saber de qué tipo fueron los primeros 3 tornillos De no saber de qué tipo de tornillos se sacaron en los 3 primeros eventos, por simetría de probabilidad, todas las posiciones son equivalentes y la probabilidad de sacar un tornillo de cobre después de 3 tornillos desconocidos es la misma que la de sacar como primer tornillo uno que sea de cobre.

Es decir, extraer sin reemplazo es lo mismo que barajar los N tornillos al azar y mirar el de la posición 4. Como el barajado es uniforme, todas las posiciones son equivalentes. La fracción de cobres en cualquier posición es $\frac{K_i}{N}$

Por ende, nos queda para cada caso particular:

- $H = A$: $\frac{0}{1000} = 0$
- $H = B$: $\frac{200}{1000} = 0.2$
- $H = C$: $\frac{250}{1000} = 0.25$
- $H = D$: $\frac{500}{1000} = 0.5$
- $H = E$: $\frac{1000}{1000} = 1$

Conociendo los tipos de tornillos que se sacaron previamente Si en las primeras j extracciones salieron r cantidad de tornillos de cobre, entonces para la extracción $j+1$ se tendrá una probabilidad de que sea de cobre de:

$$P(Cu|H_i, r, j) = \frac{K_i - r}{N - j}$$

Para cada uno de nuestros casos:

- $P(Cu|B, r, 3) = \frac{200-r}{1000-j} = \frac{200-r}{997}$
- $P(Cu|C, r, 3) = \frac{250-r}{1000-j} = \frac{250-r}{997}$
- $P(Cu|D, r, 3) = \frac{500-r}{1000-j} = \frac{500-r}{997}$

Para el caso de $H = A$ no existe probabilidad alguna ya que no existen tornillos de cobre en la caja A.

Por otro lado, para el caso $H = E$ la ecuación solo cumple si $r = j$, ya que la caja E contiene exclusivamente tornillos de cobre (por lo que sería irracional que salga otro tipo de tornillo, a menos que se plantee otro tipo de problema en la producción).

2.2 ¿Qué diferencia principal existe entre un algoritmo supervisado y un algoritmo no supervisado?

La principal diferencia entre un algoritmo supervisado y uno no supervisado, radica en la naturaleza de los datos de entrada y el objetivo del aprendizaje:

- **Algoritmo supervisado**:
  - **Datos de Entrada**: Cuentan con etiquetas (labels). Esto significa que el conjunto de datos de entrenamiento incluye tanto las entradas como las salidas correctas deseadas.
  - **Objetivo**: El problema fundamental es aprender una función que asocie las entradas con las salidas correctas. En esencia, el algoritmo es "supervisado" por las respuestas correctas.
  - **Ejemplo**: Aprendizaje Bayesiano, Knn (K nearest neighbours).

- **Algoritmo no supervisado**:
  - **Datos de Entrada**: No tienen etiquetas. El agente solo recibe los datos de entrada sin ninguna indicación de la salida correcta.
  - **Objetivo**: El problema fundamental es encontrar la estructura subyacente o patrones dentro de un conjunto de datos. El algoritmo debe inferir por sí mismo las agrupaciones o dimensiones.
  - **Ejemplo**: K-means.

3 Ejercicios de implementación

3.1 Genere un conjunto de 23 puntos que contengan coordenadas *xy* aleatorias con valores contenidos en el intervalo [0, 5] y grafique los puntos obtenidos en un gráfico.

3.1.1 Implemente un algoritmo K-means que clasifique 20 de los puntos en 2 grupos y grafique el resultado asignando un color a cada uno.

```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans

# --- 3. Generar un conjunto de 23 puntos ---
N_PUNTOS_TOTAL = 23
LIMITE_INFERIOR = 0
LIMITE_SUPERIOR = 5
N_PUNTOS_KMEANS = 20
N_CLUSTERS = 2

# Generar 23 puntos con coordenadas (x, y) aleatorias en [0, 5]
puntos_totales = np.random.uniform(low=LIMITE_INFERIOR, high=LIMITE_SUPERIOR, size=(N_PUNTOS_TOTAL, 2))

# Preparar datos para K-means
puntos_kmeans = puntos_totales[:N_PUNTOS_KMEANS]
puntos_excluidos = puntos_totales[N_PUNTOS_KMEANS:]

# --- Gráfico 1: Los 23 puntos iniciales ---
# Crea la PRIMERA figura
plt.figure(figsize=(8, 6))
plt.scatter(puntos_totales[:, 0], puntos_totales[:, 1], color='blue', label='23 Puntos Aleatorios')
plt.title('3. Puntos Aleatorios Generados en [0, 5]')
plt.xlabel('Coordenada X')
plt.ylabel('Coordenada Y')
plt.xlim(LIMITE_INFERIOR - 0.5, LIMITE_SUPERIOR + 0.5)
plt.ylim(LIMITE_INFERIOR - 0.5, LIMITE_SUPERIOR + 0.5)
plt.grid(True)
plt.legend()


# --- 3.1 Implementar un algoritmo K-means ---
kmeans = KMeans(n_clusters=N_CLUSTERS, random_state=42, n_init='auto')
kmeans.fit(puntos_kmeans)
etiquetas = kmeans.labels_
centroides = kmeans.cluster_centers_

```

```
# --- Gráfico 2: Resultado de K-means ---
# Crea la SEGUNDA figura
plt.figure(figsize=(8, 6))

# Graficar los 20 puntos clasificados
plt.scatter(puntos_kmeans[:, 0], puntos_kmeans[:, 1], c=etiquetas, cmap='cool', s=100, label=f'{N_PUNTOS_KMEANS} Puntos Clasificados (K-means)')

# Graficar los centroides
plt.scatter(centroides[:, 0], centroides[:, 1], marker='X', s=200, color='black', label='Centroides', linewidths=2)

# Graficar los puntos excluidos
plt.scatter(puntos_excluidos[:, 0], puntos_excluidos[:, 1], marker='o', s=50, color='gray', alpha=0.5, label='3 Puntos Excluidos')

plt.title(f'3.1. Clasificación K-means de {N_PUNTOS_KMEANS} Puntos en {N_CLUSTERS} Grupos')
plt.xlabel('Coordenada X')
plt.ylabel('Coordenada Y')
plt.xlim(LIMITE_INFERIOR - 0.5, LIMITE_SUPERIOR + 0.5)
plt.ylim(LIMITE_INFERIOR - 0.5, LIMITE_SUPERIOR + 0.5)
plt.grid(True)
plt.legend()

# --- Llamada UNICA a plt.show() al final ---
# Esta única llamada mostrará AMBAS figuras creadas anteriormente.
plt.show()
```

3.1.2 Tome los tres puntos restantes y clasifíquelos en los grupos obtenidos anteriormente usando el algoritmo Knn. Utilice distintos valores de K y anote lo que observa con esta elección.

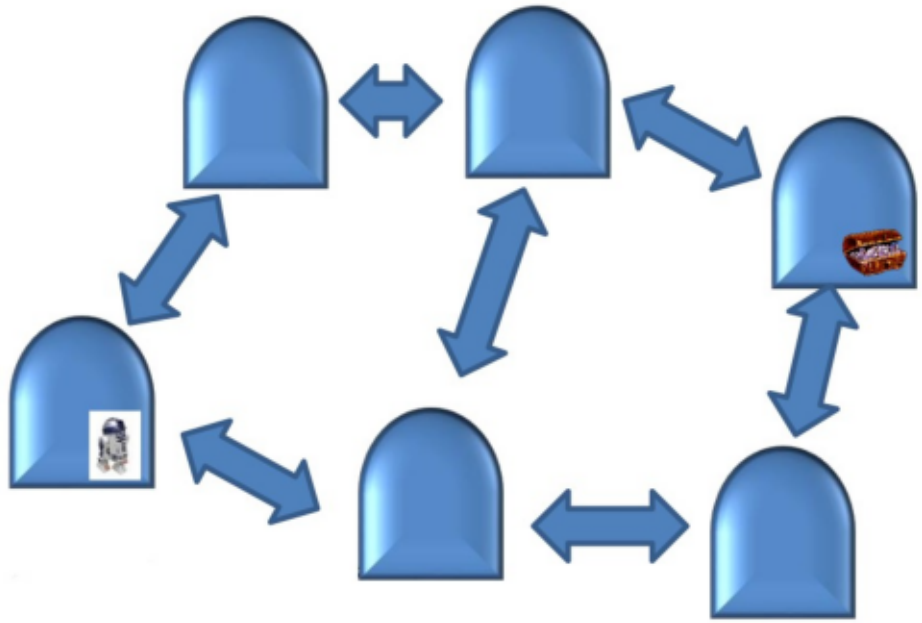
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.neighbors import KNeighborsClassifier

# --- Configuración y Generación de Datos ---
N_PUNTOS_TOTAL = 23
LIMITE_INFERIOR = 0
LIMITE_SUPERIOR = 5
N_PUNTOS_KMEANS = 20
N_CLUSTERS = 2
valores_k = [1, 3, 5] # Valores de K a probar

# Generar 23 puntos aleatorios en [0, 5]
puntos_totales = np.random.uniform(low=LIMITE_INFERIOR, high=LIMITE_SUPERIOR, size=(N_PUNTOS_TOTAL, 2))

# Preparar datos
puntos_entrenamiento = puntos_totales[:N_PUNTOS_KMEANS] # 20 puntos para K-means/K-NN entrenamiento
puntos_prueba_knn = puntos_totales[N_PUNTOS_KMEANS:]    # 3 puntos para K-NN prueba

# --- 3. Generación y Gráfico de 23 Puntos (Figura 1) ---
plt.figure(figsize=(8, 6))
```

```
plt.scatter(puntos_totales[:, 0], puntos_totales[:, 1], color='blue', label='23 Puntos Aleatorios')
plt.title('3. Puntos Aleatorios Generados en [0, 5]')
plt.xlabel('Coordenada X')
plt.ylabel('Coordenada Y')
plt.xlim(LIMITE_INFERIOR - 0.5, LIMITE_SUPERIOR + 0.5)
plt.ylim(LIMITE_INFERIOR - 0.5, LIMITE_SUPERIOR + 0.5)
plt.grid(True)
plt.legend()


# ----------------------------------------------------------------------
# --- 3.1. Implementación de K-means (Entrenamiento) ---
kmeans = KMeans(n_clusters=N_CLUSTERS, random_state=42, n_init='auto')
kmeans.fit(puntos_entrenamiento)
etiquetas_entrenamiento = kmeans.labels_ # Etiquetas (clase) de los 20 puntos
centroides = kmeans.cluster_centers_
cmap_colores = np.array(['green', 'orange']) # Color para Cluster 0 y Cluster 1

# ----------------------------------------------------------------------
# --- 3.2. Clasificación de los 3 puntos restantes con K-NN (Figura 2 con Subplots) ---
fig, axes = plt.subplots(1, 3, figsize=(18, 6), sharex=True, sharey=True)
fig.suptitle('3.2. Clasificación K-NN de 3 Puntos Restantes con Distintos Valores de K', fontsize=16)

observaciones = []
marcadores_prueba = ['D', 's', '^'] # Diamante, Cuadrado, Triángulo para los 3 puntos

for i, k in enumerate(valores_k):
    # Seleccionar el subplot actual
    ax = axes[i]

    # 1. Entrenar K-NN para el valor de K actual
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(puntos_entrenamiento, etiquetas_entrenamiento)

    # 2. Predecir las etiquetas para los 3 puntos restantes
    etiquetas_predichas = knn.predict(puntos_prueba_knn)

    # 3. Almacenar observación
    observacion_str = f"K={k}: Etiquetas predichas: {etiquetas_predichas}"
    observaciones.append(observacion_str)

    # 4. Graficar los 20 puntos de entrenamiento (Fondo)
    ax.scatter(puntos_entrenamiento[:, 0], puntos_entrenamiento[:, 1],
            c=cmap_colores[etiquetas_entrenamiento], s=80,
            label='20 Puntos Clasificados (Entrenamiento)', alpha=0.6)

    # 5. Graficar Centroides
    ax.scatter(centroides[:, 0], centroides[:, 1], marker='X', s=150, color='black', label='Centroides K-means')

    # 6. Graficar los 3 Puntos de Prueba con el color predicho
    color_predicho = cmap_colores[etiquetas_predichas]
    for j in range(len(puntos_prueba_knn)):
        ax.scatter(puntos_prueba_knn[j, 0], puntos_prueba_knn[j, 1],
                   marker=marcadores_prueba[j], s=180,
                   color=color_predicho[j], edgecolors='red', linewidths=2,
                   label=f'Punto {j+1} Predicho ({cmap_colores[etiquetas_predichas[j]]})')
```

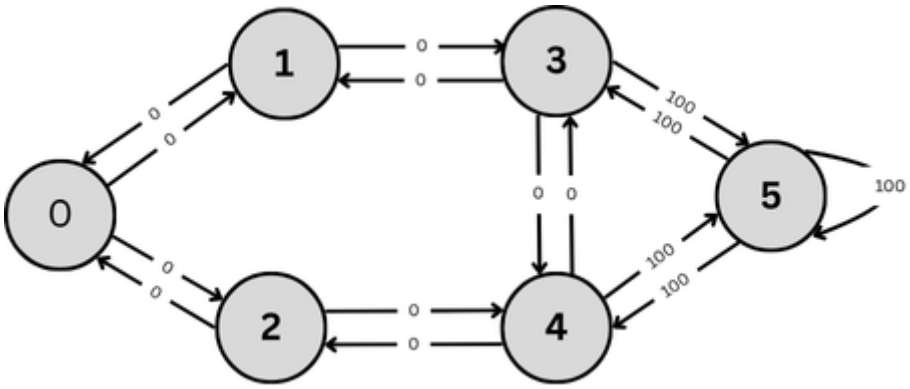
```python

    # Configurar el subplot
    ax.set_title(f'Resultado con K = {k}')
    ax.set_xlabel('Coordenada X')
    ax.set_ylabel('Coordenada Y' if i == 0 else '')
    ax.set_xlim(LIMITE_INFERIOR - 0.5, LIMITE_SUPERIOR + 0.5)
    ax.set_ylim(LIMITE_INFERIOR - 0.5, LIMITE_SUPERIOR + 0.5)
    ax.grid(True)
    ax.legend(loc='upper left', fontsize=8)


# --- Observaciones Anotadas (Sección de texto) ---
print("\n" + "=" * 50)
print("3.2. Clasificación K-NN: Resultados y Observaciones")
print("=" * 50)
print("El conjunto de entrenamiento son los 20 puntos previamente clasificados por K-means (Verde/Naranja).")
print("Los 3 puntos de prueba son clasificados según el voto de sus K vecinos más cercanos.")
print("-" * 50)
for obs in observaciones:
    print(obs)
print("-" * 50)

# Mostrar ambas figuras simultáneamente
plt.show()
```

3.2 La imagen mostrada abajo muestra una red de salas y cómo se comunican entre ellas. Implemente un algoritmo Q-Learning con un factor despreciativo $\gamma = 0.9$. La matriz de recompensas debe asignar un valor de 0 a cada camino accesible y un valor de 100 a caminos que lleven a la sala del tesoro.

```python
import requests
from PIL import Image
from io import BytesIO
import matplotlib.pyplot as plt

# URL directa de Google Drive
url = "https://drive.google.com/uc?export=view&id=1dkDruEIPa7f-BAmjI47TZl3bxaqYf6a9"

# Descargar la imagen
response = requests.get(url)
img = Image.open(BytesIO(response.content))

# Mostrar la imagen
plt.imshow(img)
plt.axis('off')  # Ocultar ejes
plt.show()
```



3.2.1 Dibuje un diagrama con la asignación de recompensas correspondiente a cada estado.



3.2.2 Diseñe y muestre la matriz de recompensas.

Matriz de recompensas

$$R = \begin{bmatrix} -1 & 0 & 0 & -1 & -1 & -1 \\ 0 & -1 & -1 & 0 & -1 & -1 \\ 0 & -1 & -1 & -1 & 0 & -1 \\ -1 & 0 & -1 & -1 & 0 & 100 \\ -1 & -1 & 0 & 0 & -1 & 100 \\ -1 & -1 & -1 & 0 & 0 & 100 \end{bmatrix}$$

3.2.3 Obtenga la matriz Q óptima, normalícela respecto al valor máximo encontrado y grafique la política obtenida en el diagrama mostrado inicialmente.

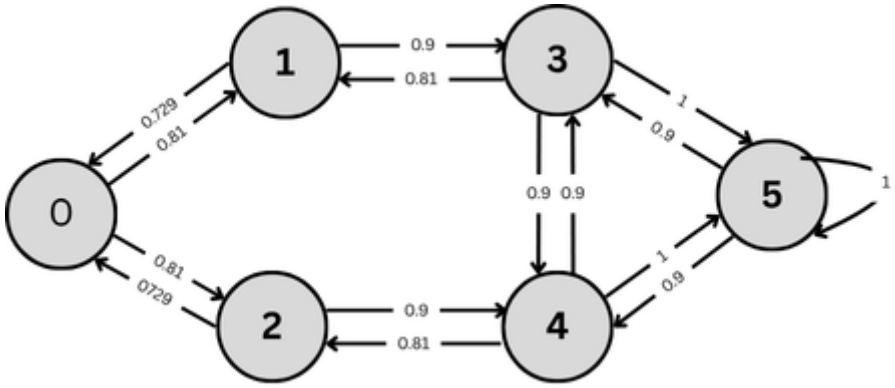
Matriz Q óptima obtenida

$$Q = \begin{bmatrix} 0 & 0.81 & 0.81 & 0 & 0 & 0 \\ 0.729 & 0 & 0 & 0.90 & 0 & 0 \\ 0.729 & 0 & 0 & 0 & 0.90 & 0 \\ 0 & 0.81 & 0 & 0 & 0.90 & 1.00 \\ 0 & 0 & 0.81 & 0.90 & 0 & 1.00 \\ 0 & 0 & 0 & 0.90 & 0.884 & 1.00 \end{bmatrix}$$

A partir de la matriz Q obtenida podemos determinar las acciones más útiles a realizar para llegar al tesoro. Iniciando en la habitación 0, vemos que los máximos valores de utilidad se dan tanto al moverse a la habitación 1 o a la 2 (0.81), ya que de ambas formas nos acercamos la misma distancia al tesoro. Luego en la habitación 1, vemos que las acciones posibles son volver a 0 o moverse a 3, siendo esta última accion la que tiene mayor valor de utilidad en este caso (0.9). De esta manera, se pueden determinar las acciones óptimas a realizar en cada estado y por lo tanto desarrollar un ruta óptima del inicio hasta el tesoro.

```
Habitación 0 -> ir a 1
Habitción 1 -> ir a 3
Habitción 2 -> ir a 4
Habitción 3 -> ir a 5
Habitción 4 -> ir a 5
Habitción 5 -> quedarse en 5
```

Gráfica de política obtenida (normalizada):



4 Anexo

4.1 Ejercicio 1

Para calcular las probabilidades de que salga cada tipo de recubrimiento electrolítico, aplicamos ley de probabilidad total:

$$P(E) = \sum P(E|H_i)P(H_i)$$

Definimos entonces: N: Recubrimiento de Niquel Cu: Recubrimiento de cobre Zn: Recubrimiento de Cincado

Luego, tenemos la distribución de "tipos de cajas". Hay 5 tipos de cajas, cada una con un tipo único de distribución de recubrimientos electrolíticos.

Tenemos para cada tipo de tornillo:

$$P(N) = \sum P(N|H)P(H)$$

$$= 1 \cdot \frac{15}{100} + 0.7 \cdot \frac{15}{100} + 0.5 \cdot \frac{50}{100} + 0.2 \cdot \frac{10}{100} + 0 \cdot \frac{10}{100}$$

$$= 0.525 = 52.5\%$$

$$P(Cu) = \sum P(Cu|H)P(H)$$

$$= 0 \cdot \frac{15}{100} + 0.2 \cdot \frac{15}{100} + 0.25 \cdot \frac{50}{100} + 0.5 \cdot \frac{10}{100} + 1 \cdot \frac{10}{100}$$

$$= 0.305 = 30.5\%$$

$$P(Zn) = \sum P(Zn|H)P(H)$$

$$= 0 \cdot \frac{15}{100} + 0.1 \cdot \frac{15}{100} + 0.25 \cdot \frac{50}{100} + 0.3 \cdot \frac{10}{100} + 0 \cdot \frac{10}{100}$$

$$= 0.17 = 17\%$$

Podríamos hacer una tabla:

Tipo	Caja A	Caja B	Caja C	Caja D	Caja E	Total
Niquel	100%	70%	50%	20%	0%	52.5%
Cobre	0%	20%	25%	50%	100%	30.5%
Cincado	0%	10%	25%	30%	0%	17%
Frecuencia Cajas	15	15	50	10	10	100

5 Bibliografía

Russell, S. & Norvig, P. (2004) *Inteligencia Artificial: Un Enfoque Moderno.* Pearson Educación S.A. (2a Ed.) Madrid, España

Poole, D. & Mackworth, A. (2017) *Artificial Intelligence: Foundations of Computational Agents.* Cambridge University Press (3a Ed.) Vancouver, Canada