Trabajo Práctico Parte 1: Análisis Bayesiano del Dataset Iris

Estadística e Inferencia II

Objetivo general:

En esta primera parte exploraremos el conjunto de datos **Iris** y aplicaremos modelos bayesianos simples y jerárquicos para modelar las características de las especies de flores. Comenzaremos con un análisis exploratorio y un modelado inicial.

NOTA: Pueden explorar otros datasets si así lo desean, indicando el lugar de donde lo obtienen y/o anexando a la entrega el archivo con los datos para poder hacer las correcciones.

Contarán con un péndice de ayuda con ejemplos de código para que tengan como plantillas. Nos guiaremos con los códigos y modelos vistos en clase, en caso de utilizar otras librerías y paquetes de Python, deben aclararlo en el informe de entrega, al igual que la bibliografía que utilicen para dar marco teórico al TP.

Parte 1: Análisis exploratorio de datos (EDA)

1. **Carga y descripción del dataset** - Cargar el dataset *Iris*. - Identificar las variables (numéricas y categóricas). - Describir brevemente cada variable (nombre, tipo y rango).

2. **Estadísticas descriptivas** - Calcular media, mediana, desviación estándar y rangos para cada variable numérica. - Realizar la tabla de frecuencia de las especies.

3. **Visualización univariada** - Graficar histogramas o densidades de cada variable numérica diferenciando por especie.

4. **Visualización bivariada** - Graficar *scatter plots* (o *pair plots*) entre pares de variables numéricas. - Colorear por especie para ver la separación entre clases.

5. **Análisis preliminar** - Identificar si hay variables más discriminantes entre especies. - Comentar relaciones visibles entre variables y especies.

Parte 2: Modelos Bayesianos

A) Modelo bayesiano no jerárquico

6. Elegir una variable numérica (por ejemplo, *Sepal.Length*).

7. Asumir que los valores de esa variable, para cada especie, provienen de una distribución Normal con media y desviación específicas para cada especie:

$$y_{ij} \sim \mathcal{N}(\mu_j, \sigma_j^2)$$

8. Especificar priors independientes para μ_j y σ_j.

9. Implementar el modelo con `PyMC` y extraer muestras posteriores de μ_j y σ_j.

10. Graficar distribuciones posteriores de las medias por especie y comparar con los valores observados.

B) Modelo bayesiano jerárquico

11. Para la misma variable, suponer que las medias de las especies provienen de una distribución común:

$$\mu_j \sim \mathcal{N}(\mu_{\text{global}}, \tau^2)$$

12. Especificar priors para μ_{global}, τ y σ_j.

13. Implementar el modelo jerárquico en `PyMC` y extraer muestras posteriores de todos los parámetros, incluyendo μ_{global} y τ.

14. Graficar las distribuciones posteriores de las medias jerárquicas y compararlas con el modelo no jerárquico.

15. Observar si ocurre *shrinkage* (medias acercándose al promedio global).

Parte 3: Comparación y conclusiones

OBSERVACIÓN: Pueden realizar una comparación de modelos utilizando las siguientes métricas de ajuste, o de forma manueal con alguna otra herramienta que consideren.

16. Calcular métricas de ajuste (WAIC o LOO) para comparar el modelo no jerárquico vs jerárquico.

17. Discutir cuál modelo se ajusta mejor a los datos y por qué.

18. (Opcional) Usar ambos modelos para predecir nuevas observaciones (posterior predictivo) y comparar resultados.

19. Resumir hallazgos del análisis exploratorio y del modelado bayesiano. Comentar ventajas y limitaciones del enfoque jerárquico en este contexto.

Apéndice de ayuda: ejemplos de código y tips

Carga y descripción del dataset

```
from sklearn.datasets import load_iris
import pandas as pd

iris = load_iris(as_frame=True).frame
iris['species'] = iris['target'].map({0:'setosa',1:'versicolor',2:'virginica'})
iris.describe()
iris['species'].value_counts()
```

Visualizaciones básicas

```
import seaborn as sns
import matplotlib.pyplot as plt

# Histograma diferenciado por especie
sns.histplot(data=iris, x='sepal length (cm)', hue='species', kde=True)
plt.show()

# Pairplot
sns.pairplot(iris, hue='species')
plt.show()
```

Modelo bayesiano no jerárquico en PyMC

```
import pymc as pm
import arviz as az
import numpy as np

# Preparar datos
species_idx = iris['species'].astype('category').cat.codes.values
y = iris['sepal length (cm)'].values
n_species = len(np.unique(species_idx))

with pm.Model() as non_hier_model:
    mu = pm.Normal('mu', mu=5.5, sigma=2, shape=n_species)
    sigma = pm.HalfNormal('sigma', sigma=1, shape=n_species)

    y_obs = pm.Normal('y_obs', mu=mu[species_idx],
                      sigma=sigma[species_idx],
                      observed=y)

    trace_non_hier = pm.sample(2000, tune=1000, target_accept=0.9)

az.plot_posterior(trace_non_hier, var_names=['mu'])
```

Modelo bayesiano jerárquico en PyMC

```
with pm.Model() as hier_model:
    mu_global = pm.Normal('mu_global', mu=5.5, sigma=2)
    tau = pm.HalfNormal('tau', sigma=1)

    mu_species = pm.Normal('mu_species', mu=mu_global, sigma=tau,
       shape=n_species)
    sigma_species = pm.HalfNormal('sigma_species', sigma=1, shape=
       n_species)

    y_obs = pm.Normal('y_obs', mu=mu_species[species_idx],
                      sigma=sigma_species[species_idx],
                      observed=y)

    trace_hier = pm.sample(2000, tune=1000, target_accept=0.9)

az.plot_posterior(trace_hier, var_names=['mu_species','mu_global','
   tau'])
```

Comparación de modelos con WAIC o LOO

```
az.compare({"no_jerarquico": trace_non_hier,
            "jerarquico": trace_hier}, ic='waic')
```

Tip: Usar `az.summary` para obtener tablas de resumen de las distribuciones posteriores.