Instituto Politécnico Nacional

Escuela Superior de Física y Matemáticas

INTRODUCCIÓN A PYMC

ALUMNA:

- MENDOZA HERNÁNDEZ MARICELA

CURSO:

- SIMULACIÓN 2

PROFESOR:

- RICARDO MEDEL ESQUIVEL

FECHA DE ENTREGA:

- 04 DE OCTUBRE 2024

Modelos Bayesianos en Python PyMC

El video nos menciona dos tipos de estadísticas, las cuales son frecuentista y bayesiana:

¿Qué es la estadística frecuentista?

Es un enfoque de la estadística que interpreta la probabilidad de un evento como la frecuencia con la que dicho evento ocurre en un gran número de experimentos o repeticiones. Se basa en la idea de que la probabilidad de un suceso es el límite de su frecuencia relativa a medida que el número de ensayos se aproxima al infinito.

Ejemplo: Si lanzamos una moneda 1,000 veces y cae "cara" en 520 ocasiones, la probabilidad frecuentista de que salga cara sería aproximadamente 0.52, ya que se ha basado en la frecuencia observada.

¿Qué es la estadística bayesiana?

Es un enfoque de la estadística que interpreta la probabilidad como un grado de creencia o confianza en un evento, en lugar de una frecuencia de ocurrencia de dicho evento. Se basa en el **Teorema de Bayes**, que permite actualizar las probabilidades iniciales (o creencias) sobre un evento a medida que se obtiene nueva evidencia o datos.
Por lo que observamos la siguiente tabla comparativa:

Aspecto	Estadística Frecuentista	Estadística Bayesiana
Fundamento	Probabilidad basada en experimentos repetidos o muestras grandes.	Probabilidad basada en el teorema de Bayes, actualización de creencias.
Enfoque	Objetivo: los parámetros de la población son fijos pero desconocidos, y las inferencias se basan en datos muestrales.	Subjetivo: los parámetros se modelan con distribuciones de probabilidad y se actualizan con nueva información.
Datos previos (conocimiento a priori)	No se considera información previa o creencias en la estimación de los parámetros.	Se utiliza una distribución a priori que refleja creencias iniciales sobre los parámetros, y se actualiza con datos nuevos.
Teorema principal	Se basa en la Ley de los Grandes Números y en procedimientos como la estimación por intervalos y pruebas de hipótesis.	Se basa en el Teorema de Bayes para actualizar probabilidades en función de nueva evidencia.
Estimación de parámetros	Los parámetros se consideran valores fijos y se estiman con muestras de datos (ej., media, varianza).	Los parámetros se modelan como distribuciones probabilísticas, actualizadas según nuevos datos.
Uso en la práctica	Más comúnmente utilizado en estudios científicos y métodos tradicionales de análisis estadístico.	Utilizado frecuentemente en situaciones de toma de decisiones, aprendizaje automático, y cuando se puede incluir conocimiento previo.

Interpretación de resultados	Los resultados son interpretados en términos de frecuencias de largo plazo y datos repetidos.	Los resultados se interpretan como grados de creencia actualizados basados en nueva evidencia.

Teorema de Bayes

El teorema de Bayes es la fórmula central de este enfoque y establece cómo se actualiza una probabilidad previa en función de la nueva evidencia, el cual está definido como:

$$P(Hipótesis\,|Datos) = \frac{P(Datos|Hipótesis) * P(Hipótesis)}{P(Datos)}$$

Donde:

- $P(Hipótesis\,|Datos)$: Es la **probabilidad posterior**, la probabilidad de la hipótesis después de observar los datos.
- $P(Datos|Hipótesis)$: Es la **verosimilitud**, la probabilidad de los datos dados que la hipótesis es cierta.
- $P(Hipótesis)$: Es la **probabilidad a priori**: la creencia inicial en la hipótesis antes de observar los datos.
- $P(Datos)$: Es la **probabilidad marginal**, la probabilidad de observar los datos, calculada considerando todas las hipótesis posibles.

Librería de Python PyMC

La librería **PyMC** nos permite definir y trabajar con **modelos bayesianos** de manera eficiente. En PyMC, podemos construir modelos probabilísticos bayesianos mediante distribuciones de probabilidad para los parámetros y los datos, para después realizar **inferencia bayesiana** sobre esos modelos usando algoritmos como el **muestreo Monte Carlo por cadenas de Márkov (MCMC)**.

El video nos explica el uso de la librería con el siguiente ejemplo:

E J E P L O

"*Es mayo del 2020, acaba de empezar la pandemia, y el gobierno quiere saber cuánta gente se ha contagiado de COVID en Santiago, Chile. Dependiendo de este número el gobierno seguirá (o no) una estrategia de inmunidad de rebaño. Para esto usaremos test que detectan anticuerpos de SARS-CoV-2.*"

- Se tomo una muestra aleatoria de 50 personas en Santiago, de los cuales 40 son negativas (nuestro test no detecta anticuerpos) y 10 son positivas (test detecta anticuerpos).

Se construyeron 3 modelos:

1. Modelo 1: Asumir que el test es perfecto.
2. Modelo 2: Incluir que el test a veces da falsos positivos.
3. Modelo 3: Incluir la incertidumbre sobre la tasa de falsos positivos.

¿Por qué se usa la distribución binomial?

Esto dado que la **distribución binomial** es adecuada cuando modelamos el número de éxitos en un conjunto de ensayos independientes (en este caso, las pruebas realizadas a las personas) con una probabilidad fija de éxito (ser positivo al test).

Para cada modelo:

- En el **Modelo 1**, puedemos asumir que la probabilidad de ser positivo es directamente la proporción estimada de la población infectada.
- En el **Modelo 2**, la distribucion binomial modela la probabilidad de que una persona sea positiva, considerando que algunos de los positivos observados son falsos.
- En el **Modelo 3**, la probabilidad de ser positivo no solo depende de la enfermedad, sino también de la incertidumbre sobre la tasa de falsos positivos, y el modelo binomial puede combinar estas incertidumbres.

Uso de librería PyMC

Despues iimplementaron los tres modelos usando la librería **PyMC** para resolver el problema paso a paso. Cada modelo abordará diferentes niveles de complejidad, desde asumir que el test es perfecto hasta incluir la incertidumbre en la tasa de falsos positivos.

PASO 1: Primero, instalemos **PyMC** y **Arviz.**

```
import pymc3 as pm
import arviz as az
```

PASO 2: Definimos nuestros datos.

```
tests_totales = 50
tests_positivos = 10
```

PASO 3: Implementamos nuestros modelos.

- ***Implementacion del modelo 1 :*** Asumimos que el test es perfecto

```
with pm.Model() as modelo_test_perfecto:
    prob = pm.Uniform(name='prob',
                      lower=0,
                      upper=1)
    casos_positivos = pm.Binomial(name='casos_positivos',
                                  p=prob,
                                  n=tests_totales,
                                  observed=tests_positivos)
    trace_test_perfecto = pm.sample(3000)
```

Se definió el modelo “modelo_test_perfecto”

Definimos la variable que queremos estimar y le damos una probabilidad a priori, se toma una distribución uniforme dado que queremos decir que todos los valores son igual de probables.

Definimos la distribución binomial y la nombramos “casoso_positivos”, en “p=prob” le pedimos a PyMC que estime la variable “prob”, por lo que modelamos el número de positivos observados (tests_positivos) como una variable aleatoria binomial con 50 ensayos y probabilidad “prob”
, finalmente en “pm.sample (3000)” realiza la inferencia usando muestreo Muestreo Monte Carlo por Cadenas de Markov.

Despues de ejecutar el código PyMC nos devuelve miles de “muestras” de nuestro parametro de interes, donde con ayuda del histogramas, vemoa que comportamiento tienen estas muestras, donde podemos ver que el 20% es nuestro valor mas probable.

```
trace_test_perfecto.get_values(varname='prop')
```

```
array([0.15966427, 0.15410507, 0.14834236, ..., 0.23318682, 0.2177276 ,
       0.18031892])
```

Ahora si nos preguntaran ¿Cuál es la probabilidad de que menos de 15% de la población se haya infectado? , con ayuda la siguiente linea del codigo lo podemos saber:


```
muestras_prop = trace_test_perfecto.get_values(varname='prob')
len(muestras_prop[muestras_prop<0.15])/len(muestras_prop)
```

```
0.13433333333333333
```

Donde le pedimos cual es la proporcion de muestras menor que 0.15, este tipo de preguntas solo se puede hacer con el modelo bayesiano.

- ***Implementación del modelo 2:*** El test a veces da falsos positivos
  “ El laboratorio que creo el test hizo 100 pruebas y nos informa que la tasa de falso positivos es de 10%”

Ahora la probabilidad de un test positivo es afectada por dos factores:

1. La probabilidad de tener COVID (lo llamaremos prov_cov)
2. La probabilidad de un falso positivo (lo llamaremos prob_fp)

La siguiente formula define lo dicho anteriormente

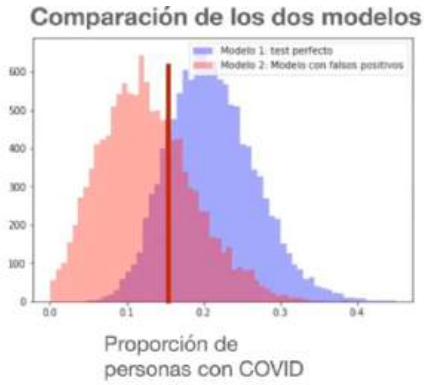
$$prob_{test_{positivo}} = prob_{cov} + (1 - prob_{cov}) * prob\_fp$$

```
with pm.Model() as modelo_test_perfecto:
    prob = pm.Uniform(name='prob',
                      lower=0,
                      upper=1)
    casos_positivos = pm.Binomial(name='casos_positivos',
                                  p=prob,
                                  n=tests_totales,
                                  observed=tests_positivos)
    trace_test_perfecto = pm.sample(3000)
```

Modelo test perfecto

```
with pm.Model() as modelo_con_fp:
    prob_cov = pm.Uniform(name='prob_cov',
                          lower=0,
                          upper=1)
    prob_fp = 0.1
    prob_test_positivo = prob_cov + (1-prob_cov)*prob_fp
    casos_positivos = pm.Binomial(name='casos_positivos',
                                  p=prob_test_positivo,
                                  n=tests_totales,
                                  observed=tests_positivos)
    modelo_con_fp = pm.sample(3000)
```

Modelo tomando en cuenta falsos positivos



Se puede observar que dado que el modelo ya sabe que existia falsos positivos , en realidad hay menos COVID de lo calculado en el modelo 1, de igual manera podemos preguntarnos , ¿Cuál es la probabilidad de que menos de 15% de la población se haya infectado?, volvemos a implementar la linea del codico usada en el modelo 1 y nos dice que:

```
muestras_prop = modelo_con_fp.get_values(varname='prob_cov')
len(muestras_prop[muestras_prop<0.15])/len(muestras_prop)
```

```
0.6703333333333333
```

Este modelo es mejor.

- Implementacion del modelo 3: No sabemos exactamente cual es la tasa de falsos positivos.

```
lab_fp_observados = 10
lab_tests_hechos = 100

with pm.Model() as modelo_con_incertidumbre:
    # Modelo para estimar la tasa de falsos positivos
    prob_fp = pm.Uniform(name='prob_fp',
                         lower=0,
                         upper=1)

    test_de_falsos_positivos = pm.Binomial(name='test_de_falsos_positivos',
                                           p=prob_fp,
                                           n=lab_tests_hechos,
                                           observed=lab_fp_observados)

    # Modelo para calcular la proporción de personas con COVID
    prob_cov = pm.Uniform(name='prob_cov',
                          lower=0,
                          upper=1)
    prob_test_positivo = prob_cov + (1-prob_cov)*prob_fp
    casos_positivos = pm.Binomial(name='casos_positivos',
                                  p=prob_test_positivo,
                                  n=tests_totales,
                                  observed=tests_positivos)
    trace_modelo_con_incertidumbre = pm.sample(3000)
```

Ahora tambien modelamos la tasa de falsos positivos con una distribucion binomial.
Se obtienen finalmente los resultados siguientes: