Instituto Politécnico Nacional

Escuela Superior de Física

y Matemáticas

Resumen: Librería PyMC

Materia: Simulación II

Profesor: Ricardo Medel Esquivel

Alumno: Zahib Gamaliel Ventura Fernandez

Modelos bayesianos en Python

Modelos bayesianos en Python

Se presenta los puntos importantes a tratar sobre este tema. La utilidad de los modelos bayesianos, ejemplos aplicados usando librerías en Python como PyMC.

Antes de empezar basta conocer acerca de la estadística bayesiana. Recordemos temas de estadística como la que se enseña en la universidad (anovas, prueba de hipótesis, ect). Este tipo de estadísticas frecuentistas piensan la probabilidad como la frecuencia que pasa un evento. Por ejemplo, si lanzamos una moneda unas mil veces cual será la frecuencia de caras.

Los bayesianos pensamos la probabilidad como una forma de medir incertidumbre. Genera soluciones más intuitivas.

Este tipo de estadística parte del teorema de bayes

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Este teorema nos dice: la probabilidad condicional de un evento aleatorio A dado B en términos de la distribución de probabilidad condicional del evento B dado A y la distribución de probabilidad marginal de solo A. De una forma sencilla, $P(A|B)$ es lo que queremos calcular (conocimiento posterior), $P(B|A)$ es la probabilidad de ver los datos que observados en distintos casos hipotéticos (likelihood) y $P(A)$ también llamado el conocimiento a priori.

Realizar los cálculos de esta probabilidad puede ser un tanto complicado y algunas veces tardado. Para eso ocuparemos de Python la librería PyMC la cual necesita el conocimiento a priori y likelihood para estimar un modelo bayesiano, por lo que nos evita hacer la matemática compleja.

Veamos un ejemplo, 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.

A partir de aquí aleatoricemos y hagamos una encuesta. Recordemos la distribución binomial, hay N experimentos en éxito (1) o en fracaso (0) con probabilidad de éxito p.

En este caso podemos considerar el hecho de que al aplicarle una persona una prueba para detectar si tiene COVID, los resultados pueden ser positivo o negativo, sea éxito al resultado de que una persona tenga COVID y fracaso a que no tenga. La probabilidad p es la que queremos estimar.

Tomaremos una muestra aleatoria de 50 personas en Santiago; de las 50 pruebas, 40 son negativas (nuestra prueba no detecta anticuerpos) y 10 positivas (la prueba si detecta anticuerpos).

En base a estos datos se construirán 3 modelos:

Modelo 1. Asume que la prueba es perfecta.

Modelo 2. La prueba a veces da falsos positivos.

Modelo 3. Incluiremos la incertidumbre sobre la tasa de falsos positivos.

De los 3 modelos el mas interesante es el modelo 3, que pasa cuando no tenemos el número exacto de la tasa de falsos positivos. Debemos poner esa incertidumbre en el modelo, es donde aquí los modelos bayesianos destacan.

Antes de empezara a modelar estos datos, lo primero es preguntar: ¿Cómo modelar estos datos? Puesto que son N experimentos y hay éxito o fracaso con probabilidad p (es la proporción de gente con COVID) es un ejemplo de una distribución binomial, cada prueba es un experimento.

Asumiendo que la prueba es perfecta **(Modelo 1),** el código del modelo es a continuación:

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

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

```
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)
```

La primera sección del código muestra las librerías a utilizar, en la imagen se importa la librería PyMC3, se debe a versiones anteriores mas ahora es PyMC y la librería arviz, esta ultima complementa a la primera para visualizar datos, gráficos.

La segunda sección del código se definen nuestros datos, recordemos que nos 50 pruebas y 10 salieron en éxito.

En la última sección del código se define la variable que queremos estimar (en este caso p) y le damos una probabilidad a priori. Definimos la distribución de los datos el llamado (likelihood). Hay que notar que en la línea de código donde [prob=pm.Uniform(name=

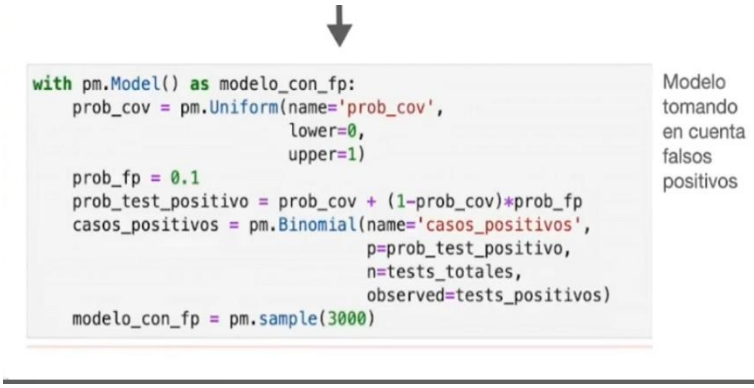
’prob’,lower=0,upper=1)], considerando [p = prom] es la variable que queremos estimar. Por lo que PyMC lo hace y nada más.

La última línea de código nos saca 3000 muestras lo que también es el posterior según el teorema de bayes, pero aplicando estadística bayesiana.

La prueba da falsos positivos **(Modelo 2).**

Consideremos que el laboratorio que creo las pruebas hizo 100 y nos informa que la tasa de falsos positivos es del 10%. De las 10 pruebas, no sabemos el número exacto, esperemos alrededor de 5 falsos positivos, en nuestro modelo bayesiano esto es fácil de incorporar diciéndole al modelo que la proporción de pruebas positivas no es lo mismo que la gente con COVID.

Ahora la probabilidad de una prueba positiva es afectada por dos factores: la probabilidad de tener COVID y la probabilidad de un falso positivo. De esto obtenemos la fórmula de la relación de ambos [prob_test_positivo = prob_cov + (1 – prob_cov)*prob_falso_pov]. El código es el siguiente:



```
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

Con un par de líneas más al código del modelo 1 podemos calcular la probabilidad de pruebas positivas.

Por ultimo la incertidumbre sobre la tasa de falso positivo **(Modelo 3).**

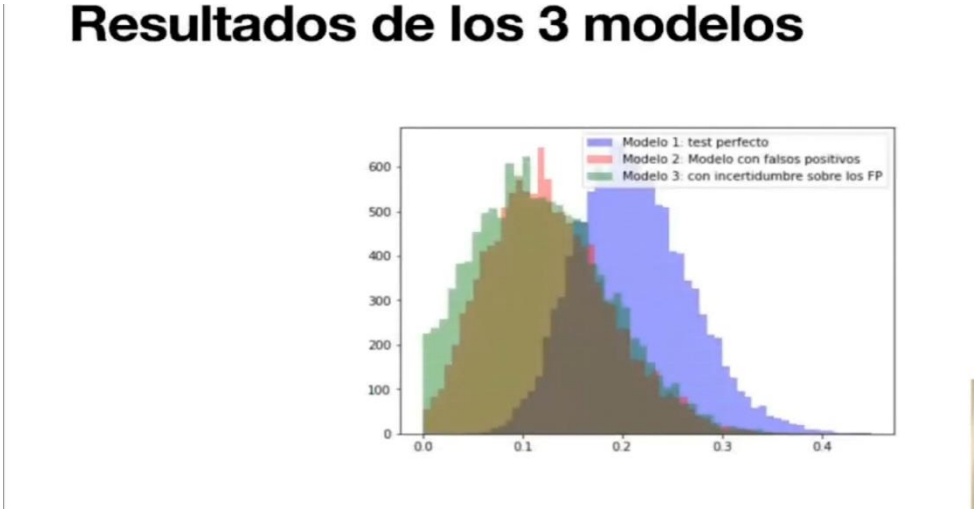
Podemos pensar que la tasa de falso positivos también es un parámetro que tenemos que estimar, ¿podemos seguir usando la distribución binomial como los casos anteriores?

Código a continuación:

```
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 también modelamos la tasa de falso positivo con una distribución binomial. Observemos que basta con agregar otras líneas de cogido y PyMC hace toda la matemática compleja por nosotros lo que nos llevaría a varias horas de trabajo y muchos dolores de cabeza.

Una comparación del histograma de los tres modelos



Recapitulando

- Hicimos modelos muy robustos sin tener que usar matemática compleja
- Los hicimos de manera iterativa
- Pudimos contestar preguntas practicas que un modelo frecuentista no puede hacerlo
- Pudimos incorporar toda la incertidumbre