yz709-ds-sup1

Question 1
Question 2
Question 4
Question 5
Question 6
Question 7
Question 8

Question 1

Question 1. Given a dataset $(x_1, \ldots, x_n)$, we wish to fit a Poisson distribution. This is a discrete random variable with a single parameter $\lambda > 0$, called the rate, and

$$\Pr(x\,;\,\lambda) = \frac{\lambda^x e^{-\lambda}}{x!} \quad \text{for } x \in \{0, 1, 2, \ldots\}.$$

Show that the maximum likelihood estimator for λ is $\hat{\lambda} = n^{-1} \sum_{i=1}^{n} x_i$.

$(x_1, \ldots, x_n)$

$\Pr(x; \lambda) = \dfrac{\lambda^x e^{-\lambda}}{x!} \quad x \in \{0, 1, 2, \ldots\}$

$\log \Pr(x; \lambda) = x \log(\lambda) - \lambda - \log(x!)$

$\text{loglik} = \sum_{i=1}^{n} \log \Pr(x_i; \lambda) = \sum_{i=1}^{n} x_i (\log(\lambda)) - \lambda n - \sum_{i=1}^{n} \log(x_i!)$

when $\dfrac{\partial\, \text{loglik}}{\partial \lambda} = \dfrac{\sum_{i=1}^{n} x_i}{\lambda} - n = 0$

$\Rightarrow \hat{\lambda} = n^{-1} \sum_{i=1}^{n} x_i$

Comments:

Question 2

Question 2. Given a dataset [3,2,8,1,5,0,8], we wish to fit a Poisson distribution. Give code to achieve this fit, using `scipy.optimize.fmin`.

$X \sim \text{Poi}(\lambda)$, $\lambda > 0$

$\Pr(x; \lambda) = \frac{\lambda^x e^{-\lambda}}{x!}$

since scipy.optimize.fmin needs parameters $\in \mathbb{R}$

let $\lambda = e^t$, $t \in \mathbb{R}$

```
x = np.array(3, 2, 8, 1, 5, 0, 8)

def loglik(x, t):
  return np.log(scipy.stats.poisson.pmf(x, t))

t_hat = scipy.optimize.fmin(lambda l: -np.sum(loglik(x, np.log(l))), np.log(0.2))
l_hat = np.exp(t_hat) # l_hat is the optimised lambda value
```

💡 Comments:

Question 4

Question 4. Given a dataset $(x_1, \ldots, x_n)$, we wish to fit the Uniform$[0, \theta]$ distribution, where θ is unknown. Show that the maximum likelihood estimator is $\hat{\theta} = \max_i x_i$.

$$\mathbb{P}(X \le x) = \begin{cases} 0 & , x < 0 \\ \frac{x}{\theta} & , x \in [0, \theta] \\ 1 & , x > \theta \end{cases}$$

$$\Pr(x;\theta) = \frac{d}{dx}\mathbb{P}(X \le x) = \begin{cases} 0 & , x < 0 \\ \frac{1}{\theta} & , x \in [0, \theta] \\ 0 & , x > \theta \end{cases}$$

$$= \frac{1}{\theta} I_{x \geq 0} I_{\theta \geq x}$$

$$\prod_{i=1}^{n} \Pr(x_i;\theta) = \frac{1}{\theta} \prod_{i=1}^{n} I_{x_i \geq 0} I_{\theta \geq x_i}$$

hence at $\max_i x_i$, all indicator functions would be 1, hence $\prod_{i=1}^{n} \Pr(x_i;\theta) = \frac{1}{\theta}$, where $\theta \geq \max_i x_i$

since $\frac{1}{\theta}$ decreases as θ increases, we select the smallest $\theta \in [\max_i x_i, \infty)$

$\Rightarrow \hat{\theta} = \max_i x_i$

Comments:

Question 5

Question 5 (A/B testing). Your company has two systems which it wishes to compare, A and B. It has asked you to compare the two, on the basis of performance measurements $(x_1, \ldots, x_m)$ from system A and $(y_1, \ldots, y_n)$ from system B. Any fool using Excel can just compare the averages, $\bar{x} = m^{-1} \sum_{i=1}^{m} x_i$ and $\bar{y} = n^{-1} \sum_{i=1}^{n} y_i$, but you are cleverer than that and you will harness the power of Machine Learning.

Suppose the x_i are drawn from $X \sim \text{Normal}(\mu, \sigma^2)$, and the y_i are drawn from $Y \sim \text{Normal}(\mu + \delta, \sigma^2)$, and all the samples are independent, and μ, δ, and σ are unknown. Find maximum likelihood estimators for the three unknown parameters.

Q5

$$\Pr(x;\mu,\sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

$$\Pr(y;\mu,\delta,\sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-(\mu+\delta))^2}{2\sigma^2}}$$

$$\text{loglik\_x} = \sum_{i=1}^{m} \log \Pr(x_i;\mu,\sigma) = -\frac{m}{2}\log(2\pi\sigma^2) - \sum_{i=1}^{m}\frac{(x_i-\mu)^2}{2\sigma^2}$$

$$\text{loglik\_y} = \sum_{i=1}^{n} \log \Pr(y_i;\mu,\delta,\sigma) = -\frac{n}{2}\log(2\pi\sigma^2) - \sum_{i=1}^{n}\frac{(y_i-(\mu+\delta))^2}{2\sigma^2}$$

$$\frac{\partial\, \text{loglik\_x}}{\partial\mu} = \sum_{i=1}^{m}\frac{x_i-\mu}{\sigma^2} = 0 \qquad \frac{\partial\, \text{loglik\_x}}{\partial\sigma} = -\frac{m}{2\pi\sigma} + \sum_{i=1}^{m}\frac{(x_i-\mu)^2}{\sigma^3} = 0$$

$$\Rightarrow \mu = \frac{\sum_{i=1}^{m} x_i}{m} = \bar{x} \qquad \sigma^2 = \frac{2\pi}{m^2}\sum_{i=1}^{m}(x_i-\bar{x})^2$$

$$\frac{\partial\, \text{loglik\_y}}{\partial\mu} = \sum_{i=1}^{n}\frac{y_i-\mu-\delta}{\sigma^2} = 0 \qquad \Rightarrow \sigma = \frac{\sqrt{2\pi}}{m}\sqrt{\sum_{i=1}^{m}(x_i-\bar{x})^2}$$

$$\Rightarrow \mu = \frac{\sum_{i=1}^{n} y_i - \delta}{n} = \bar{y} - \frac{\delta}{n}$$

$$\Rightarrow \delta = \sum_{i=1}^{n} y_i - n\bar{x}$$

Comments:

Question 6

Question 6. Let x_i be the population of city i, and let y_i be the number of crimes reported. Consider the model $Y_i \sim \text{Poisson}(\lambda x_i)$, where $\lambda > 0$ is an unknown parameter. Find the maximum likelihood estimator $\hat{\lambda}$.

$Y_i \sim \text{Poisson}(\lambda x_i)$, $\lambda > 0$

$$\Pr(y; \lambda x_i) = \frac{(\lambda x_i)^y e^{-\lambda x_i}}{y!}$$

$$\log \Pr(y; \lambda x_i) = y \log(\lambda x_i) - \lambda x_i - \log(y!)$$

$$\text{loglik} = \sum_{j=1}^{n} \log \Pr(y_j; \lambda x_i) = [\log(\lambda) + \log(x_i)] \sum_{j=1}^{n} y_j - \lambda n x_i - \sum_{j=1}^{n} \log(y_j!)$$

$$\frac{\partial\, \text{loglik}}{\partial \lambda} = \frac{\sum_{j=1}^{n} y_j}{\lambda} - n x_i = 0$$

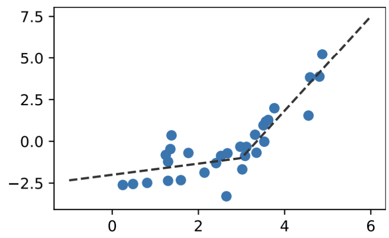
$$\hat{\lambda} = x_i \bar{y}$$

population of city i ↲ ↳ the average number of crimes

Comments:

Question 7

Question 7. We wish to fit a piecewise linear line to a dataset, as shown below. The inflection point is given, and we wish to estimate the slopes and intercepts. Explain how to achieve this using a linear modelling approach.

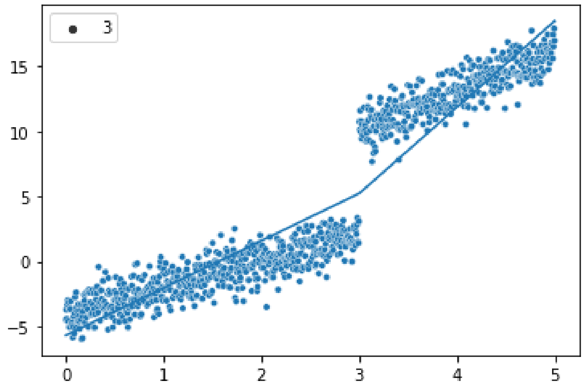


*Note. As a sanity check, you should implement your model formula as a function and plot it. Here's a function that **fails** the check.*

```
def pred(x, m₁,c₁,m₂,c₂, inflection_x=3):
    e = numpy.where(x <= inflection_x, 1, 0)
    return e*(m₁*x + c₁) + (1−e)*(m₂*x+c₂)
x = numpy.linspace(0,5,1000)
plt.plot(x, pred(x, m₁=0.5,c₁=0,m₂=1,c₂=2))
```

$$y = \beta_0 + \beta_1 x + \beta_2 (x-3) 1_{x \geq 3}$$

- When $x < 3$, we have $y = \beta_0 + \beta_1 x$ because $1_{x \geq 3} = 0$
- When $x \geq 3$, we have $y = \beta_0 + \beta_1 x + \beta_2 (x-3) = (\beta_0 - 3\beta_1) + (\beta_1 + \beta_2) x$




```
import scipy, seaborn as sns
def f(β, x, y, knot):
  β0,β1,β2 = β
  return np.sum(np.power(y-(β0+β1*x+β2*(x-knot)*np.where(x>=3,1,0)),2))
def pred(β, x, knot):
  β0,β1,β2 = β
  return β0+β1*x+β2*(x-knot)*np.where(x>=3,1,0)

x = np.linspace(0,5,1000)
y = np.array([np.where(i>=3, 3 * i + 1 + np.random.normal(0, 1), 2 * i - 4 + np.random.normal(0, 1)) for i i
n x])
β = scipy.optimize.fmin(lambda β: f(β, x, y, 3), (1,1,1), maxiter=100)

sns.scatterplot(x,y,size=3)
sns.lineplot(x, pred(β,x,3))
plt.show()
```

💡 Comments:

Question 8

Question 8. For the climate data from section 2.2.5 of lecture notes, we proposed the model

$$\texttt{temp} \approx \alpha + \beta_1 \sin(2\pi\texttt{t}) + \beta_2 \cos(2\pi\texttt{t}) + \gamma\texttt{t}$$

in which the $+\gamma\texttt{t}$ term asserts that temperatures are increasing at a constant rate. We might suspect though that temperatures are increasing non-linearly. To test this, we can create a non-numerical feature out of `t` by

```
u = 'decade_' + str(math.floor(t/10)) + '0s'
```

(which gives us values like `'decade_1980s'`, `'decade_1990s'`, etc.) and fit the model

$$\texttt{temp} \approx \alpha + \beta_1 \sin(2\pi\texttt{t}) + \beta_2 \cos(2\pi\texttt{t}) + \gamma_{\texttt{u}}.$$

Write this as a linear model, and give code to fit it. *[Note. You should explain what your feature vectors are, then give a one-line command to estimate the parameters.]*

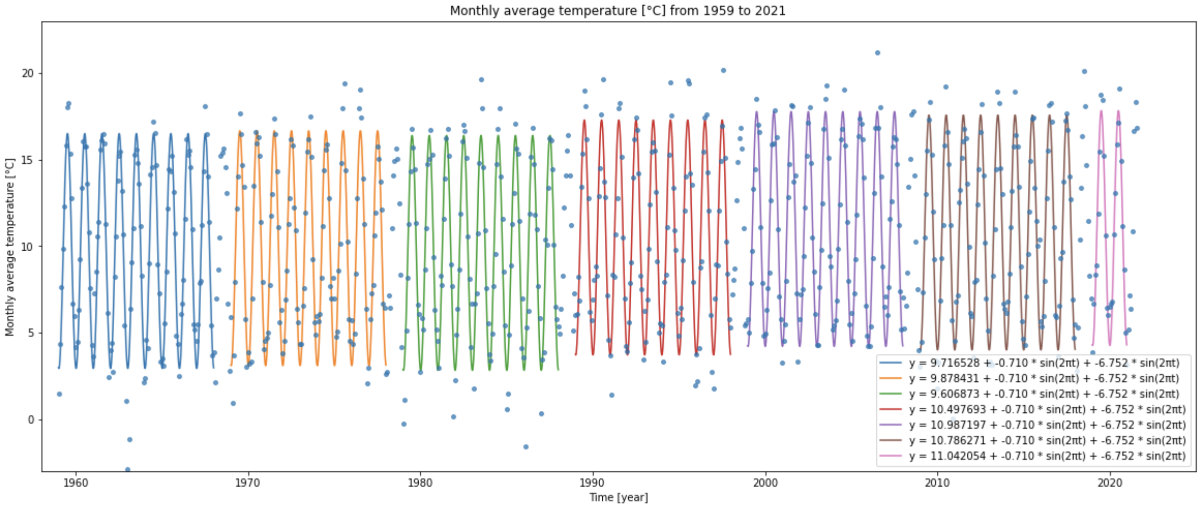
- Set up a base line model with no Secular Trend: y = α + β1 * sin(2πt) + β2 * sin(2πt)
- Assume we add on a variable β3 which changes every 10 years
- Then split the years into [1959-1968], [1969-1978], [1979-1988], [1989-1998], [1999-2008], [2009-2018], [2019-2021], each with a linear model y = α + β1 * sin(2πt) + β2 * sin(2πt) + β3 = (a+β3) + β1 * sin(2πt) + β2 * sin(2πt), where β3 is different and all other coefficients are the same and the feature vectors are `[np.ones(len(t), np.sin(2*π*t), np.cos(2*π*t)]`

```
# a base linear model which uses all data from year 1959 to year 2021
# y = α + β1 * sin(2πt) + β2 * sin(2πt)
t,y = data[0]
X = np.column_stack([np.sin(2*π*t), np.cos(2*π*t)])
model = sklearn.linear_model.LinearRegression()
model.fit(X, y)
coef_base,(β1,β2) = (model.intercept_, model.coef_)

coef = []
pred = []

# minimising error term squared
def f(v, x_val, y_val):
  return np.sum(np.power(y_val - (v + β1 * np.sin(2*π*x_val) + β2 * np.cos(2*π*x_val)),2));

for x,y in data:
  new_α = scipy.optimize.fmin(lambda v : f(v, x, y), coef_base, maxiter=100);  # find the optimal coefficient α
  coef.append((new_α[0],β1,β2))
  t = np.linspace(np.min(x), np.max(x), 1000)
  Xnew = np.column_stack([np.sin(2*π*t), np.cos(2*π*t)])
  t_pred = new_α + β1 * np.sin(2*π*t) + β2 * np.cos(2*π*t);
  pred.append((t, t_pred))
```




```
# output:
"""
from year 1959 to year 1968, coef difference to the base line model (β3): 0.000
from year 1969 to year 1978, coef difference to the base line model (β3): 0.162
from year 1979 to year 1988, coef difference to the base line model (β3): -0.110
from year 1989 to year 1998, coef difference to the base line model (β3): 0.781
from year 1999 to year 2008, coef difference to the base line model (β3): 1.271
from year 2009 to year 2018, coef difference to the base line model (β3): 1.070
from year 2019 to year 2021, coef difference to the base line model (β3): 1.326
"""
```

💡 Comments: