1 Example Sheet 1

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

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

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

Since the throws are independent of each other, the probability of all of them happening with a parameter λ is given by:

$$\begin{aligned}\Pr((x_1, \ldots, x_n)\,|\lambda) &= \prod_{i=1}^{n} \Pr(x_i|\lambda) \\ &= \prod_{i=1}^{n} \frac{\lambda^{x_i} e^{-\lambda}}{x_i!} \\ &= \frac{\lambda^{\sum_{i=1}^{n} x_I} e^{-n\lambda}}{\prod_{i=1}^{n} x_i!}\end{aligned}$$

The maximum likelihood estimator $\hat{\lambda}$ for this can be found by differentiating with respect to λ and finding the maximum.
With $S = \sum_{i=1}^{n} x_i$ and $P = \prod_{i=1}^{n} x_i!$

$$\begin{aligned}\frac{\mathrm{d}\Pr((x_1, \ldots, x_n)\,|\lambda)}{\mathrm{d}\lambda} &= \frac{S\lambda^{S-1}e^{-n\lambda} - n\lambda^S e^{-n\lambda}}{P} \\ &= \frac{(S - n\lambda)\lambda^{S-1}e^{-n\lambda}}{P}\end{aligned}$$

Setting this equal to 0 gives the equation:

$$\begin{aligned}&0 = \frac{(S - n\lambda)\lambda^{S-1}e^{-n\lambda}}{P} \implies \\ &0 = (S - n\lambda)\lambda^{S-1}e^{-n\lambda} \implies \\ &0 = S - n\lambda \vee 0 = \lambda^{S-1}e^{-n\lambda} \implies \\ &\lambda = \frac{S}{n} \vee \hat{\lambda} = 0\end{aligned}$$

By definition of the Poisson distribution, $\lambda > 0$ hence $\hat{\lambda} \neq 0$. We can therefore conclude that the maximum likelihood estimator $\hat{\lambda} = \frac{S}{n} = \frac{\sum_{i=1}^{n} x_i}{n}$ as required.

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.

For this and all following code fragments I will assume the following imports:

```
import numpy as np
import pandas as pd
import scipy.stats as stats
import scipy.optimize as opt
```

The following code finds the optimum value for the parameter. The value found is 3.857153320312502 and matches the value found using the mathematical method.

```
data = [3,2,8,1,5,0,8]

def logprob(x):
        return np.sum(stats.poisson.logpmf(data,np.exp(x)))

param = np.exp(opt.fmin(lambda x: -logprob(x),1))
```

3. The dataset in question 2 comes from counts of radioactive particle emissions. The technician says that the counter's display is defective and that any value larger than 20 just displays as 20. You therefore decide to model the datapoints as a truncated Poisson distribution,

$$\Pr(x;\lambda) = \begin{cases} \frac{\lambda^x e^{-\lambda}}{x!} & \text{if } x \in \{0,1,\dots,19\} \\ 1 - \sum_{r=0}^{19} \frac{\lambda^r e^{-\lambda}}{r!} & \text{if } x = 20 \\ 0 & \text{if } x > 20 \end{cases}$$

(a) Your engineer friends thinks one should use unbiased estimators rather than maximum likelihood estimators. Show that for the Poisson probability model in question 1 the maximum likelihood estimator $\hat{\lambda} = \frac{\sum_{i=1}^n x_i}{n}$ is unbiased.

Let X be distributed with a Poisson distribution with parameter $\frac{\sum_{i=1}^n}{n}$. $\mathbb{E}(X) = \frac{\sum_{i=1}^n}{n}$. So the expectation of n samples from this distribution is $n\mathbb{E}(X) = n\frac{\sum_{i=1}^n}{n} = \sum_{i=1}^n x_i$. This is the observed mean. Hence the maximum likelihood estimator $\hat{\lambda}$ is also unbiased.

(b) Explain why, for the dataset in question 2, $\hat{\lambda} = \frac{\sum_{i=1}^n x_i}{n}$ is the maximum likelihood estimate for the truncated Poisson model.

All the data in the dataset $[3, 2, 8, 1, 5, 0, 8]$ are strictly less than 20. For this range of probability the truncated Poisson distribution has the same probability density as the Poisson distribution. Therefore the equation and derivation of the maximum likelihood estimator is the same.

(c) Show that $\hat{\lambda} = \frac{\sum_{i=1}^n}{n}$ is *not* an unbiased estimator in the truncated Poisson model.

Let X be distributed according to the truncated Poisson distribution with parameter $\frac{\sum_{i=1}^n x_i}{n}$

$$\begin{aligned} \mathbb{E}(X) &= \sum_{x=0}^{\infty} x \cdot \Pr(X = x) \\ &= x \sum_{x=0}^{20} x \cdot \Pr(X = x) \\ &= 20 \cdot \left(1 - \sum_{r=0}^{19} \frac{\lambda^r e^{-\lambda}}{r!}\right) + \sum_{x=0}^{19} \frac{\lambda^x e^{-\lambda}}{x!} \\ &< \sum_{x=0}^{\infty} \frac{\lambda^x e^{-\lambda}}{x!} \\ &< \frac{\sum_{i=1}^n x_i}{n} \implies \end{aligned}$$

Therefore the observed mean is greater than the mean of X with parameter $\frac{\sum_{i=1}^n}{n}$. Therefore we can conclude that $\frac{\sum_{i=1}^n}{n}$ is not an unbiased estimator in the truncated Poisson model.

(d) Which do you think one should use, maximum likelihood estimators or unbiased estimators. Why?

In the general case, I thin maximum likelihood estimators are best. They're often easier to calculate numerically and usually give better results. Unbiased estimators are not unique, many of them are very poor and have high bias. Worse, in many discrete distributions, unbiased estimators don't exist.

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

The probability density function for parameter X distributed with the Uniform$[0, \theta]$ distribution (I will assume the distribution is discrete) is given by:

$$\Pr(X = x) = \begin{cases} \frac{1}{\theta+1} & \text{if } x <= \theta \\ 0 & \text{otherwise} \end{cases}$$

With $1_{\theta \geq x_i}$ as the indicator function for $\theta \geq x_i$; the probability of observing the data $[3, 2, 8, 1, 5, 0, 8]$ is given by:

$$\begin{aligned} \Pr((x_1, \ldots, x_n)) &= \prod_{i=1}^{n} \frac{1}{\theta} 1_{\theta \geq x_i} \\ &= \frac{1}{\theta^n} \prod_{i=1}^{n} 1_{\theta \geq x_i} \\ &= \frac{1}{\theta^n} 1_{\theta \geq \max_i x_i} \end{aligned}$$

Since the probability decreases as θ increases for all $\theta \geq \max_i x_i$ and is 0 for $\theta < \max_i x_i$, we can see that the maximum probability occurs at $\theta = \max_i x_i$.

5. 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, $\bar{x} = \frac{\sum_{i=1}^m x_i}{m}$ and $\bar{y} = \frac{\sum_{i=1}^n x_i}{n}$, but you are cleverer than that and will harness the power of Machine Learning.

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

Using the probability density function for the Normal distribution, we can write down the probability of observing the x_i and y_i. We shall then take the partial derivative by all variables at the same time and find the value for which all the partial derivatives are 0 and verify this value is a maxima. These will give us maximum likelihood estimators.

$$\begin{aligned} \Pr((x_1, \ldots, x_m) \wedge (y_1, \ldots, y_n)) &= \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x_i-\mu)^2}{2\sigma^2}} \times \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(y_i-\mu-\delta)^2}{2\sigma^2}} \\ &= \frac{1}{\left(\sqrt{2\pi}\sigma\right)^{m+n}} e^{-\frac{\sum_{i=1}^m (x_i-\mu)^2 + \sum_{i=1}^n (y_i-\mu-\delta)^2}{2\sigma^2}} \end{aligned}$$

$$\begin{aligned}
\frac{\partial \Pr}{\partial \mu} &= \frac{\left(\sum_{i=1}^{m}(x_i-\mu)+\sum_{i=1}^{n}(y_i-\mu-\delta)\right)}{\sigma^2}\frac{1}{\left(\sqrt{2\pi}\sigma\right)^{m+n}}e^{-\frac{\sum_{i=1}^{m}(x_i-\mu)^2+\sum_{i=1}^{n}(y_i-\mu-\delta)^2}{2\sigma^2}} \\
&= \frac{\sum_{i=1}^{m}x_i+\sum_{i=1}^{n}y_i-(m+n)\mu-n\delta}{\sigma^2}\frac{1}{\left(\sqrt{2\pi}\sigma\right)^{m+n}}e^{-\frac{\sum_{i=1}^{m}(x_i-\mu)^2+\sum_{i=1}^{n}(y_i-\mu-\delta)^2}{2\sigma^2}} \\
0 &= \sum_{i=1}^{m}x_i+\sum_{i=1}^{n}y_i-(m+n)\mu-n\delta
\end{aligned}$$

$$\begin{aligned}
\frac{\partial \Pr}{\partial \delta} &= \frac{\sum_{i=1}^{n}(y_i-\mu-\delta)}{\sigma^2}\frac{1}{\left(\sqrt{2\pi}\sigma\right)^{m+n}}e^{-\frac{\sum_{i=1}^{m}(x_i-\mu)^2+\sum_{i=1}^{n}(y_i-\mu-\delta)^2}{2\sigma^2}} \\
&= \frac{\sum_{i=1}^{n}y_i-n\delta-n\mu}{\sigma^2}\frac{1}{\left(\sqrt{2\pi}\sigma\right)^{m+n}}e^{-\frac{\sum_{i=1}^{m}(x_i-\mu)^2+\sum_{i=1}^{n}(y_i-\mu-\delta)^2}{2\sigma^2}} \\
0 &= \sum_{i=1}^{n}y_i-n\delta-n\mu
\end{aligned}$$

$$\begin{aligned}
\frac{\partial \Pr}{\partial \sigma} &= \left(\frac{\sum_{i=1}^{m}(x_i-\mu)^2+\sum_{i=1}^{n}(y_i-\mu-\delta)^2}{\sigma^3}-\frac{m+n}{\sigma}\right)\frac{1}{\left(\sqrt{2\pi}\sigma\right)^{m+n}}e^{-\frac{\sum_{i=1}^{m}(x_i-\mu)^2+\sum_{i=1}^{n}(y_i-\mu-\delta)^2}{2\sigma^2}} \\
0 &= \sum_{i=1}^{m}(x_i-\mu)^2+\sum_{i=1}^{n}(y_i-\mu-\delta)^2-(m+n)\sigma^2
\end{aligned}$$

We can subtract the second expression from the first to get:

$$\begin{aligned}
0-0 &= \sum_{i=1}^{m}x_i+\sum_{i=1}^{n}y_i-m\mu-n\mu-n\delta+n\delta+n\mu-\sum_{i=1}^{n}y_i \\
0 &= \sum_{i=1}^{m}x_i-m\mu \\
\mu &= \frac{\sum_{i=1}^{m}x_i}{m}
\end{aligned}$$

We can then substitute this result into the second expression to get:

$$\begin{aligned}
0 &= \sum_{i=1}^{n}y_i-n\delta-n\mu \\
0 &= \sum_{i=1}^{n}y_i-n\delta-\frac{n\sum_{i=1}^{m}x_i}{m} \\
n\delta &= \sum_{i=1}^{n}y_i-\frac{n\sum_{i=1}^{m}x_i}{m} \\
\delta &= \frac{\sum_{i=1}^{n}y_i}{n}-\frac{\sum_{i=1}^{m}x_i}{m}
\end{aligned}$$

Substituting these values into the expression for σ gives:

$$\begin{aligned}
0 &= \sum_{i=1}^{m}(x_i-\mu)^2 + \sum_{i=1}^{n}(y_i-\mu-\delta)^2 - (m+n)\sigma^2 \\
(m+n)\sigma^2 &= \sum_{i=1}^{m}\left(x_i - \frac{\sum_{i=1}^{m} x_i}{m}\right)^2 + \sum_{i=1}^{n}\left(y_i - \frac{\sum_{i=1}^{n} y_i}{n}\right)^2 \\
\sigma^2 &= \frac{\sum_{i=1}^{m}\left(x_i - \frac{\sum_{i=1}^{m} x_i}{m}\right)^2 + \sum_{i=1}^{n}\left(y_i - \frac{\sum_{i=1}^{n} y_i}{n}\right)^2}{m+n}
\end{aligned}$$

6. Let x_i be the population of city $i \in \{1, \ldots, n\}$, 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 $\hat{\lambda}$.

   Assuming that the crimes in each city are unrelated to each other, the probability of observing the number of crimes we have is given by:

$$\begin{aligned}
\Pr((y_1,\ldots,y_n)|\lambda) &= \prod_{i=1}^{n} \frac{(\lambda x_i)^{y_i}\, e^{-\lambda x_i}}{y_i!} \\
&= \lambda^{\sum_{i=1}^{n} y_i} e^{-\lambda \sum_{i=1}^{n} x_i} \prod_{i=1}^{n} \frac{x_i^{y_i}}{y_i!}
\end{aligned}$$

   We can differentiate this with respect to λ and solve for 0 derivative to find the maximum likelihood estimator $\hat{\lambda}$.

$$\begin{aligned}
\frac{\partial \Pr}{\partial \lambda} &= \left(\lambda^{\left(\sum_{i=1}^{n} y_i\right)-1} \sum_{i=1}^{n} y_i - \lambda^{\left(\sum_{i=1}^{n} y_i\right)} \sum_{i=1}^{n} x_i\right) e^{-\lambda \sum_{i=1}^{n} x_i} \prod_{i=1}^{n} \frac{x_i^{y_i}}{y_i!} \\
&= \left(\sum_{i=1}^{n} y_i - \lambda \sum_{i=1}^{n} x_i\right) \lambda^{\left(\sum_{i=1}^{n} y_i\right)-1} e^{-\lambda \sum_{i=1}^{n} x_i} \prod_{i=1}^{n} \frac{x_i^{y_i}}{y_i!} \\
0 &= \sum_{i=1}^{n} y_i - \hat{\lambda} \sum_{i=1}^{n} x_i \\
\hat{\lambda} &= \frac{\sum_{i=1}^{n} y_i}{\sum_{i=1}^{n} x_i}
\end{aligned}$$

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.

   The equation of a line which passes through a point (x', y') can be given by: $y = m(x - x') + y'$. We must fit two of those. This can be done with an indicator variable, two gradients and a one-hot encoding. Note that we have no intercepts – since we know a point the line passes through, we know the intercept.

$$y \approx m_1(x-x')1_{x \le x'} + m_2(x-x')1_{x > x'} + y'$$

```
"""
xp, yp is the inflection point
x, y are the x and y coordinates of the data
"""
```

```
def pred(m1, m2):
        e = x <= xp
        return m1*(x-xp)*e+m2*(x-xp)*np.invert(e)+yp

def error(v):
        np.sum(np.power(pred(*v) - y, 2))

opt.fmin(error ,[0 ,0])
```

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

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

in which the $+\gamma t$ 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"

and fit the model

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

Write this as a linear model, and give code to fit it.

The linear is:

$$\text{temp} \approx \beta_1 \sin(2\pi t) + \beta_2 \cos(2\pi t) + \gamma_{1950s} 1_{1950s} + \cdots + \gamma_{2020s} 1_{2020s} + \mathcal{N}(0, \sigma)$$

Where γ_{XYZ0s} is the average temperature in decade XYZ, and 1_{XYZ0s} is the indicator variable as to whether this temperature is in the decade XYZ.

```
url = "https://.../climate.csv"
climate = pd.read_csv(url)
df = climate.loc[(climate.station == "Cambridge")]
t = df.yyyy + (df.mm - 1) / 12
temp = (df.tmin + df.tmax) / 2
decades = np.array([t // 10 == i for i in range(195, 203)])

def predict(x):
        return x[0]*np.sin(2*np.pi*t)+x[1]*np.cos(2*np.pi*t)+ x[2:-1]@decades

def logprob(x):
        return np.sum(stats.norm.logpdf(predict(x), temp, np.exp(x[-1])))

opt.fmin(lambda x: -logprob(x), [-1,-7,10,10,10,10,10,10,10,10,0.3])
```

9. I have two feature vectors

$$\text{gender} = [f, f, f, f, m, m, m] \text{eth} = [a, a, b, w, a, b, b]$$

and I one-hot encode them as

$$g_1 = [1,1,1,1,0,0,0] \qquad e_1 = [1,1,0,0,1,0,0]$$
$$g_2 = [0,0,0,0,1,1,1] \qquad e_2 = [0,0,1,0,0,1,1]$$
$$e_3 = [0,0,0,1,0,0,0]$$

Are these five vectors $\{g_1, g_2, e_1, e_2, e_3\}$ linearly independent? If not, find a linearly independent set of vectors that spans the same feature space.

The vectors $\{g_1, g_2, e_1, e_2, e_3\}$ are not linearly independent. The following dependencies exist:

$$g_2 = \mathbf{1} - g_2$$
$$e_3 = \mathbf{1} - e_1 - e_2$$

We can remove these dependencies by considering only the set of $\{g_1, g_2, e_1, e_2, e_3\}$.

So the vectors we use are:

$$g_1 = [1,1,1,1,0,0,0] \qquad e_1 = [1,1,0,0,1,0,0]$$
$$e_2 = [0,0,1,0,0,1,1]$$

10. For the police stop-and-search dataset in section 2.6, we wish to investigate intersectionality in police bias. We propose the linear model

$$1[\text{outcome} = \text{“find”}] \approx \alpha_{\text{gender}} + \beta_{\text{eth}}$$

Write this as a linear model using one-hot coding. Are the parameters identifiable? If not, rewrite the model so they are and interpret the parameters of your model.

$$1[\text{outcome} = \text{“find”}] \approx \alpha_1 g_1 + \beta_1 e_1 + \beta_2 e_2 + \gamma$$

The features are identifiable. γ is the rate for a white male. α_1 is the difference between the male gender and the female gender, β_1 is the difference between white ethnicity and asian ethnicity and β_2 is the difference between white ethnicity and black ethnicity.

2 Supplementary Questions

11 Fit the model

$$\text{Petal.length} \approx \alpha - \beta(\text{Sepal.length})^{\gamma}, \gamma > 0$$

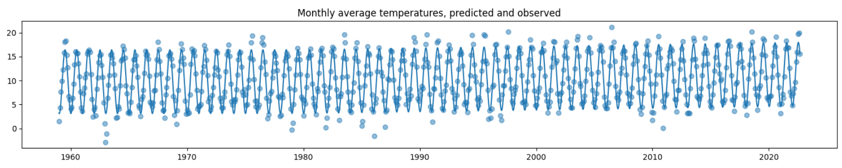
by minimizing the mean square error.

```
url = "https://.../iris.csv"
iris = pd.read_csv(url)

def pred(x):
        return x[0] - x[1] * np.power(iris['Sepal.Length'], x[2])

def mean_squared_error(x):
        return np.sum(np.power(iris['Petal.Length'] - pred(x), 2))

opt.fmin(mean_squared_error, [-160000, -160000, 0.00007])
```

12 As an alternative to the model from question 8, we might suspect that temperatures are increasing linearly up to 1980, and that they are increasing linearly at a different rate from 1980 onwards. Devise a linear model to express this, using your answer to question 7, and fit it. Plot your fit.

$$\text{temp} \approx \alpha + \beta_1 \sin(2\pi t) + \beta_2 \cos(2\pi t) + \gamma_1 t 1_{t<=1980} + \gamma_2 t 1_{t>1980}$$

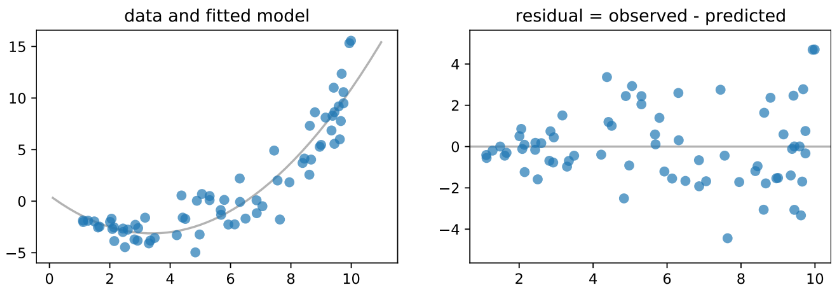
```
url = "https://.../climate.csv"
climate = pd.read_csv(url)
df = climate.loc[(climate.station == "Cambridge")]
t = df.yyyy + (df.mm - 1) / 12
temp = (df.tmin + df.tmax) / 2

def predict(x):
        return x[0] + x[1] * np.sin(2*np.pi*t) + x[2] * np.cos(2*np.pi*t) + \
        x[3] * (t <= 1980) + x[4] * (t > 1980)

def logprob(x):
        return np.sum(stats.norm.logpdf(temp, predict(x), np.exp(x[-1])))

opt.fmin(lambda x: -logprob(x), [12, -1, -7, -2, -1, 0.4])
```



13 We are given a dataset with predictor x and label y and we fit the linear model:

$$y_i \approx \hat{\alpha} + \hat{\beta} x_i + \hat{\gamma} x_i^2$$

After fitting the model using least squares estimation, we plot the residuals $\epsilon_i = y_i - (\hat{\alpha} + \hat{\beta} x_i + \hat{\gamma} x_i^2)$.



(a) Describe what you would expect to see in the residual plot, if the assumptions behind linear regression are correct.

I would expect residual error to be uncorrelated with x.

(b) This residual plot suggests that perhaps $\epsilon_i \sim \mathcal{N}(0, (\sigma x_i)^2)$ where σ is an unknown parameter. Assuming this is the case, give pseudocode to find the maximum likelihood estimators for α, β and γ.

```
url = "https://.../heteroscedasticity.csv"
hetero = pd.read_csv(url)

def predict(p):
        return p[0] + p[1] * hetero.x + p[2] * hetero.x ** 2

def probability(p):
        return np.sum(stats.norm.logpdf(hetero.y, predict(p), p[3] * hetero.x))

opt.fmin(lambda x: -probability(x), [0, -2, 1, 0.5])
```

14 Let $(F_1, F_2, F_3, \dots) = (1, 1, 2, 3, \dots)$ be the Fibonacci numbers, $F_n = F_{n-1} + F_{n-2}$. Define the vectors f, f_1, f_2 and f_3 by

$$\begin{aligned} f &= [F_4, F_5, F_6, \dots, F_{m+3}] \\ f_1 &= [F_3, F_4, F_5, \dots, F_{m+2}] \\ f_2 &= [F_2, F_3, F_4, \dots, F_{m+1}] \\ f_3 &= [F_1, F_2, F_3, \dots, F_m] \end{aligned}$$

for some large value of m. If you were to fit the linear model

$$f \approx \alpha + \beta f_1 + \beta f_2$$

what parameters would you expect?

Because of the relationship $F_n = F_{n-1} + F_{n-2}$, the feature vectors are not independent. Specifically $f = f_1 + f_2$.

So if we were to fit the model $f \approx \alpha + \beta f_1 + \beta f_2$, I would expect the parameters to be $\alpha = 0, \beta_1 = 1, \beta_2 = 1$.

What about the linear model

$$f \approx \alpha + \beta f_1 + \beta f_2 + \beta_3 f_3$$

There are two dependencies here. $f = f_1 + f_2$ and $f_1 = f_2 + f_3$. Therefore the features are not linearly independent and so there is no unique solution. However, for any solution to be optimum, the following equations must hold: $\alpha = 0, \beta_1 + \beta_3 = 1, \beta_2 - \beta_3 = 1$.

15 For the police stop-and-search data from section 2.6, consider the model

$$1[\text{outcome} = \text{find}] = \alpha + \sum_{k \neq \text{White}} \beta_k (1[\text{eth} = k] - 1[\text{eth} = \text{White}])$$

Interpret the parameters.

- α is the probability of a white person being stop-and-searched

- β_k is the probability of a person of ethnicity $k \neq$ White being stop-and-searched

16 Sketch the cumulative distribution functions for these two random variables. Are they discrete or continuous?

```
def rx():
        u = random.random()
        return 1/u

def ry():
        u2 = random.random()
        return rx() + math.floor(u2)
```

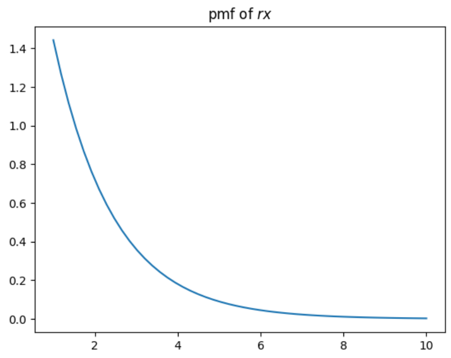
- *rx*

  rx is a continuous random variable with the following probability density function:

$$P(rx = x) = \begin{cases} 0 & \text{if } x < 1 \\ \frac{2}{\ln 2} 2^{-x} & \text{if } x \geq 1 \end{cases}$$

  Notice first that the probability that $rx < 2$ is equal to the probability that $u \geq \frac{1}{2}$. Then notice the general case $P\left(rx < 2^x\right) = P\left(u > 2^{-x}\right) = 1 - 2^{-x}$. We can then differentiate this general expression to get the probability density function (note we only consider it over the valid domain $rx \geq 1$):

$$\begin{aligned} \text{pmf}(X) &= \frac{\partial P(X < x)}{\partial x} \\ &= \frac{2}{\ln 2} 2^{-x} \end{aligned}$$



- *ry*

  ry is a continuous random variable with the following probability density function:

$$P(ry = y) = \begin{cases} 0 & \text{if } y < 1 \\ \frac{2}{\ln 2} 2^{-y} & \text{if } y \geq 1 \end{cases}$$

  Note that this is the same as rx. math.floor(u2) will take the floor of u2 – for all $u2 < 1$ this will be zero. However, the probability $u2 \geq 1$ is equal to the probability $u2 = 1$ – which is 0 as $u2$ is a continuous distribution. Therefore the probability that math.floor($u2$) = 1 is also zero. Therefore ry has the same distribution as rx.

17 Is it possible for a continuous random variable to have a probability density function that approaches ∞ at some point in the support? Is it possible to have this and also have finite mean and variance?

It is possible for a continuous random variable to have a probability density function approaching ∞ for some point. It is also possible for this variable to have finite mean and variance.

Consider the continuous random variable X with probability density function:

$$\operatorname{pdf}_X(x) = \begin{cases} \frac{1}{2\sqrt{x}} & \text{if } 0 \le x \le 1 \\ 0 & \text{otherwise} \end{cases}$$

Note firstly that $\lim_{x\to 0} \operatorname{pdf}_X(x) = \infty$.

To prove pdf_X is a valid probability density function, we have to show it integrates to 1.

$$\begin{aligned} \int_0^1 \operatorname{pdf}_X(x)\,\mathrm{d}x &= \int_0^1 \frac{1}{2\sqrt{x}}\,\mathrm{d}x \\ &= \left[\sqrt{x}\right]_0^1 \\ &= 1 \end{aligned}$$

So this is a valid pdf which approaches ∞ at some point.

The mean is finite:

$$\begin{aligned} \bar{x} &= \int_0^1 x \cdot \operatorname{pdf}_X(x)\,\mathrm{d}x \\ &= \int_0^1 \frac{1}{2} \cdot x\,\mathrm{d}x \\ &= \frac{1}{2} \end{aligned}$$

So the mean mean of X is $\frac{1}{2}$ – which is finite.

The variance is finite:

$$\begin{aligned} \operatorname{Var}(x) &= \int_0^1 x^2 \cdot \operatorname{pdf}_X(x)\,\mathrm{d}x\,\mathrm{d}x - \hat{x}^2 \\ &= \int_0^1 \frac{1}{2} \cdot x^2\,\mathrm{d}x\,\mathrm{d}x - \frac{1}{2}^2 \\ &= \left[\frac{1}{2} \cdot x\right]_0^1 - \frac{1}{4} \\ &= \frac{1}{4} \end{aligned}$$

So the variance of X is $\frac{1}{4}$ – which is finite.

Therefore it is possible for a variable to have a pdf approaching ∞ at some point, and to have a finite mean and variance.

3 2018 Paper 6 Question 8

https://www.cl.cam.ac.uk/teaching/exams/pastpapers/y2018p6q8.pdf

Fisher's Iris dataset contains, among other things, measurements of Petal .Length and Sepal.Length for samples from each of three species of iris. Suppose we want to fit the model

$$\text{Petal.Length} = \alpha_s + \beta_s \text{Sepal.Length} + \mathcal{N}(0, \sigma^2)$$

where s is the species.

(a) Explain what is meant by "linear model", "feature" and "orthogonal projection". Rewrite the above model as a linear model made up of linearly independent features and explain why they are linearly independent.

A linear model is a model of the form $y_i \approx \sum_{i=1}^{n} k_i x_i$ for some response y_i, parameters k_i and features x_i. Intuitively, a linear model is a model which can be rewritten as a matrix multiplication.

A feature is an input variable. In the context of data science, a feature is a dimension in the input vector.

An orthogonal projection is a projection of a response vector $\mathbf{y}$ into a feature space $(\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_n)$ such that all the features are linearly independent and there exist constants $(\alpha_1, \alpha_2, \ldots, \alpha_n)$ such that $\mathbf{y} = \sum_{i=1}^{n} \alpha_i f_i$.

$$\text{Petal.Length} \approx \sum_{s=1}^{n} (\alpha_s 1_{s=\text{species}} + \beta_s \text{Sepal.Length} 1_{s=\text{species}})$$

The features above are linearly independent because no feature is a linear combination of any of the other features. For this particular dataset, you can determine $1_{s=\text{species}}$ from Sepal.Length$1_{s=\text{species}}$, however this is not a linear relationship and so the features are linearly independent.

(b) You are given a library function proj$(y, [e_1, \ldots, e_n])$. It returns a list $[\lambda_1, \ldots, \lambda_n]$ such that $\lambda_1 e_1 + \cdots + \lambda_n e_n$ is the orthogonal projection of the vector y onto the subspace spanned by vectors $\{e_1, \ldots, e_n\}$. Explain what is meant by the "least squares method" and give pseudocode using proj to find the least squares estimators for α_s and β_s.

The "least squares method" minimises the sum of the squares of the residuals (the differences between the prediction and the response). Least squares can either be done mathematically or computationally.

```
url = "https://.../iris.csv"

iris = pd.read_csv(url)

# if we use set then the order is not guai
species = list(sorted(set(iris['Species'])))

response = iris['Petal.Length']
features = []
for each in species:
        features.append(iris['Species'] == each)
        features.append((iris['Species'] == each) * df['Sepal.Length'])

features = np.array(features)

params = proj(response, features)
```

```
var = np.sum(np.power(features @ params - response, 2)) / response.shape[0]

sigma = np.sqrt(var)
```

The least squares estimator for $\alpha_{\text{species[i]}}$ is params[2 * i].
The least squares estimator for $\beta_{\text{species[i]}}$ is params[2 * i + 1].
The least squares estimator for σ is sigma.

(c) Explain how to compute the maximum likelihood estimators of α_s, β_s and σ. In your answer, you should explain the relationship between the least squares method and maximum likelihood estimation.

The mathematical method:

To find the maximum likelihood estimators for α_s, β_s and σ we need to form an expression for the probability of observing the data that we did in terms of the parameters α_s, β_s and σ. We should then form n equations by differentiating with respect to each of the parameters. This can then be solved like a simultaneous equation and will form a closed form expression for each parameter. It may be easier to use the logarithmic probability as it will remove all the exponents created by the normal distribution. Note that we must optimise over all parameters at the same time. If we do not, we will end up with equations in terms of other parameters.

$$
\begin{aligned}
\Pr &= \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{\left(sum_{s=1}^{n} \alpha_s 1_{s=\text{species}} + \beta_s \text{Sepal.Length} 1_{s=\text{species}} - \text{Petal.Length}\right)^2}{2\sigma^2}} \\
\ln \Pr &= \sum_{i=1}^{n} \left(\ln \frac{1}{\sqrt{2\pi}\sigma} - \frac{\left(sum_{s=1}^{n} \alpha_s 1_{s=\text{species}} + \beta_s \text{Sepal.Length} 1_{s=\text{species}} - \text{Petal.Length}\right)^2}{2\sigma^2} \right) \\
&= -\frac{n}{2} \ln 2\pi - n \ln \sigma - \left(\ln \frac{1}{\sqrt{2\pi}\sigma} - \frac{\left(sum_{s=1}^{n} \alpha_s 1_{s=\text{species}} + \beta_s \text{Sepal.Length} 1_{s=\text{species}} - \text{Petal.Length}\right)^2}{2\sigma^2} \right)
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial \ln \Pr}{\partial \alpha_s} &= \dots \\
\frac{\partial \ln \Pr}{\partial \beta_s} &= \dots \\
\frac{\partial \ln \Pr}{\partial \sigma} &= \dots
\end{aligned}
$$

Then combine these equations and solve. The results will be the maximum likelihood estimators for α_s, β_s and σ.

However, for many difficult datasets or complicated models, the maths can become unmanageable. In these cases, it is often easier to determine maximum likelihood estimators computationally.

The computational method:

Form an expression for the probability of our model generating the data that has been observed in terms of α_s, β_s and σ. Then use a numerical optimization function (such as scipy.optimize.fmin) to find the values of the parameters that maximise this probability. Note that we can use a function that finds the minimum to find the maximum by minimizing the negative. In almost all cases, we must use the logarithmic probabilities – as the raw probability tends to zero very quickly (leading to floating point precision errors). The parameters returned will be the maximum likelihood estimators.

```
response = iris['Petal.Length']
features = []
for each in species:
        features.append(iris['Species'] == each)
        features.append((iris['Species'] == each) * df['Sepal.Length'])

features = np.array(features)

def predict(x):
        return features @ x

def logprob(x):
        return np.sum(stats.norm.logpdf(response, predict(x[:-1]), x[-1]))

opt.fmin(lambda x: -logprob(x), [...])
```

Finding the parameters such that the product of the normal probability density functions between the predicted values and the response probabilities is minimised (probability is maximised) is equivalent to finding parameters that minimise the square of the difference between the response and the prediction. Therefore least squares is equivalent to maximum likelihood estimation over a normal distribution. If the residuals are distributed with any distribution other than normal, this equivalence does not necessarily hold.

Assume we have some response vector $\mathbf{y}$ and predictions $\mathbf{x}$:

$$\begin{aligned}
\Pr &= \prod \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(\mathbf{x}-\mathbf{y})^2}{2\sigma^2}} \\
\ln \Pr &= \sum -\frac{1}{2}\ln 2\pi - \ln\sigma - \frac{(\mathbf{x}-\mathbf{y})^2}{2\sigma^2} \\
&= -\frac{n}{2}\ln 2\pi - n\ln\sigma - \sum \frac{(\mathbf{x}-\mathbf{y})^2}{2\sigma^2}
\end{aligned}$$

Maximising probability is equivalent to minimising the negative of the logarithmic probability:

$$\ln \Pr = \frac{n}{2}\ln 2\pi + n\ln\sigma + \sum \frac{(\mathbf{x}-\mathbf{y})^2}{2\sigma^2}$$

If we partially differentiate $\mathbf{x}$, we notice that the minima of σ. Therefore maximising the probability is equivalent to minimising $\sum(\mathbf{x}-\mathbf{y})^2$ – the least squares method

Therefore, provided the residuals are distributed normally, the least squares method is equivalent to maximum likelihood estimation.

(d) We wish to know whether the β_s coefficients for the three species are noticeably different. Outline the Bayesian approach to answering the question.

The Bayesian approach is to build a probability density function for the distribution of the parameters. This distribution can be formed using Bayes rule with a prior belief about the distribution of the parameters and the observed results.

Critically, the distribution should be a joint probability distribution of *all* parameters. Additionally, we should be very careful not to make the probability that a parameter has a certian value zero without certainty that it is not that value. The prior distribution becomes less and less important as we gather more data; however if the probability of a parameter having a certain value is zero in the prior then it will remain zero throughout.

Bayes' rule states that:

$$\Pr(\beta; X) = \frac{p(\beta)\Pr(X;\beta)}{\Pr(X)}$$

- $\Pr(\beta; X)$ is the probability distribution of β given the observed data X
- $p(\beta)$ is the prior belief of the distribution of β
- $\Pr(X; \beta)$ is the probability of the observed data given X
- $\Pr(X)$ is the probability of the observed data occurring

This can be easily generalised to form a probability density function for multiple parameters.

$$\begin{aligned}
\Pr(\alpha, \beta; X) &= \frac{p(\alpha)p(\beta;\alpha)\Pr(X;\alpha,\beta)}{\Pr(X)} \\
\Pr(\alpha, \beta, \sigma; X) &= \frac{p(\alpha)p(\beta;\alpha)p(\sigma;\alpha,\beta)\Pr(X;\alpha,\beta,\sigma)}{\Pr(X)} \\
&\dots
\end{aligned}$$

The Bayesian approach would approximate this distribution for all parameters. The probability that the β_s are the same is formed by reparameterising the β_s into a single parameter β and then integrating. IE the following equation:

$$p_{\text{same}} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{0}^{\infty} \Pr(\beta, \alpha_{s_1}, \alpha_{s_2}, \alpha_{s_3}, \sigma)\,\mathrm{d}\sigma\,\mathrm{d}\alpha_{s_3}\,\mathrm{d}\alpha_{s_2}\,\mathrm{d}\alpha_{s_1}\,\mathrm{d}\beta$$

The solution can be approximated computationally for example using scipy.integrate or by a discrete approximation.

Integrating over 5 variables simultaneously is computationally difficult and so in practice we would use a discrete approximation (computational Bayes) or simplify the equation further.