exercise 2

fitting probabilistic models

solutions due

until **November 19, 2023** at **23:59** via **ecampus**

general remarks

see previous exercise sheets

task 2.1 [10 points]

bivariate Gaussian models and conditional expectations

The file

`whData.dat`

contains the body height and -weight data we know from the lectures. To read it into memory (i.e. into numpy arrays), you could proceed as follows

```
import numpy as np

data = np.loadtxt('whData.dat', dtype=np.object, comments='#', delimiter=None)
w = data[:,0].astype(float)
h = data[:,1].astype(float)
```

task 2.1.1 [2 points] pre-processing

Observe that the data contains two outliers (with a body weight of -1) since two of the surveyed students did not disclose how much they weigh.

Implement numpy code that removes the outliers and gathers the remaining data points in an array which represents a data matrix

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}_1 & \boldsymbol{x}_2 & \cdots & \boldsymbol{x}_n \end{bmatrix}$$

where

$$\boldsymbol{x}_j = \begin{bmatrix} h_j \\ w_j \end{bmatrix}$$

task 2.1.2 [2 points] model fitting

Use maximum likelihood parameter estimation to fit a bivariate Gaussian distribution to the data in $\boldsymbol{X}$. That is, use standard numpy functionalities (i.e. the functions *mean* and *cov*) to compute the parameter estimates

$$\hat{\boldsymbol{\mu}} = \frac{1}{n} \sum_j \boldsymbol{x}_j$$

$$\hat{\boldsymbol{\Sigma}} = \frac{1}{n-1} \sum_j \left[\boldsymbol{x}_j - \hat{\boldsymbol{\mu}}\right] \left[\boldsymbol{x}_j - \hat{\boldsymbol{\mu}}\right]^\intercal$$

Note: MLE of the parameters of a Gaussian is "essentially" a deterministic procedure. While working with different computers / operating systems / software versions may lead to ever so slightly different results, it would be very surprising if your team came up with estimates $\hat{\boldsymbol{\mu}}$ and $\hat{\boldsymbol{\Sigma}}$ that would substantially differ from those your instructors computed.

task 2.1.3 [6 points] prediction making

Given your fitted Gaussian model, implement the expression for computing the conditional expectation $\mathbb{E}\big[w \mid h\big]$ and use it to predict body weights for body heights of $h \in \big\{140, 150, 160, 170, 180, 190, 200, 210\big\}$.

What do you observe? Given your "everyday experience", would you say your results are plausible?

Ponder the question of whether or not "plausibility" is something we could easily quantify or measure ...

task 2.2 [20 points]

fitting a Weibull distribution to a histogram (part 1)

In our lectures, we frequently work with Gaussian models because they are exceptionally easy to handle. However, ease of use should never be our primary objective when modeling real life data. Instead, models must be plausible and be able to capture crucial characteristics of the data we are working with even if this means that model fitting may be burdensome. In this task, we shall explore this.

task 2.2.1

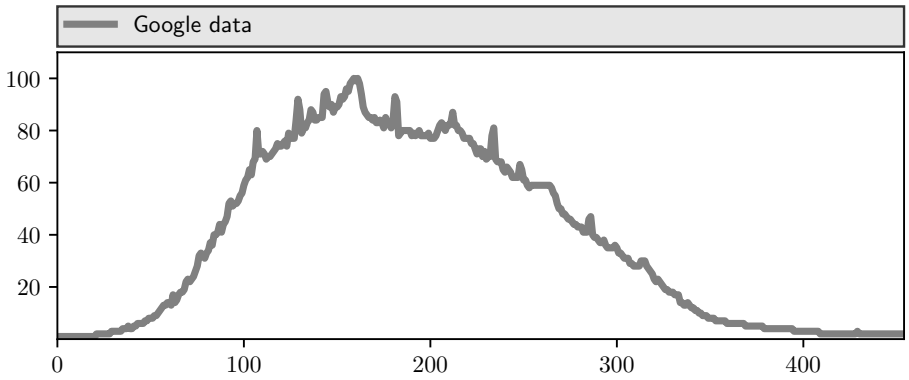
The file

`myspace.csv`

contains Google Trends data which indicate how global interest in the query term *"myspace"* evolved over time.

Read the data in the second column of this file and remove leading zeros! Store the remaining entries in an array $\boldsymbol{h} = [h_1, h_2, \ldots, h_n]$ and create an array $\boldsymbol{t} = [1, 2, \ldots, n]$.

If you would plot this histogram data $h[t]$ (you do not have to), you should get something like this



This figure clearly shows that Google searches for *myspace* were not Gaussian distributed. Over time, interest first grew exponentially, reached some peak activity, and then declined about linearly.

In other words, while a Gaussian distribution is symmetric, the distribution in the figure is positively skewed (its right tail is longer than its left tail). Modeling the evolution of searches for *myspace* in terms of a Gaussian would thus be inappropriate.

One more thing: the figure also shows that Google does not disclose true search volumes. Rather, Goolge Trends data is always scaled such that the maximum search activity corresponds to a value of 100. From the point of view of (probabilistic) model fitting this is unfortunate but provides us with an opportunity of getting used to real life situations ...

task 2.2.2 [20 points]

Now, fit a Weibull distribution to the histogram $h[t]$. The probability density function of the Weibull distribution is given by

$$f(\,t \mid \alpha, \beta) = \tfrac{\alpha}{\beta}\big(\tfrac{t}{\beta}\big)^{\alpha-1}\, e^{-\left(\frac{t}{\beta}\right)^{\alpha}} \tag{1}$$

where α and β are a shape- and a scale parameter.

Note: please do *not* work with the Weibull distributions in `scipy.stats`. These are lobotomized version of the distribution in (1) and nobody knows why the scipy developers opted to ignore the real Weibull distribution ...

Note: to this day, neither `scipy.stats` nor `scipy.optimize` ship with functionalities for fitting distributions to histograms. Again, nobody knows why but this provides another learning opportunity! Please implement the following approach.

Unlike for the Gaussian distribution, maximum likelihood estimation of the parameters of a Weibull distribution is more involved.

Given a data sample $D = \big\{d_i\big\}_{i=1}^{N}$, the log-likelihood for the parameters of the Weibull distribution is

$$L(\alpha, \beta \mid D) = N\big(\log\alpha - \alpha\log\beta\big) + (\alpha-1)\sum_i \log d_i - \sum_i \Big(\frac{d_i}{\beta}\Big)^{\alpha}$$

Deriving L with respect to α and β leads to a coupled system of partial differential equations for which there is no closed form solution. Therefore, resort to Newton's method for iterative simultaneous equation solving and compute

$$\begin{bmatrix}\alpha\\ \beta\end{bmatrix} \leftarrow \begin{bmatrix}\alpha\\ \beta\end{bmatrix} + \begin{bmatrix}\frac{\partial^2 L}{\partial\alpha^2} & \frac{\partial^2 L}{\partial\alpha\partial\beta}\\ \frac{\partial^2 L}{\partial\alpha\partial\beta} & \frac{\partial^2 L}{\partial\beta^2}\end{bmatrix}^{-1} \begin{bmatrix}-\frac{\partial L}{\partial\alpha}\\ -\frac{\partial L}{\partial\beta}\end{bmatrix}$$

where the entries of the gradient vector and the Hessian matrix amount to

$$\frac{\partial L}{\partial\alpha} = \frac{N}{\alpha} - N\log\beta + \sum_i \log d_i - \sum_i \left(\frac{d_i}{\beta}\right)^\alpha \log\frac{d_i}{\beta}$$

$$\frac{\partial L}{\partial\beta} = \frac{\alpha}{\beta}\left(\sum_i \left(\frac{d_i}{\beta}\right)^\alpha - N\right)$$

$$\frac{\partial^2 L}{\partial\alpha^2} = -\frac{N}{\alpha^2} - \sum_i \left(\frac{d_i}{\beta}\right)^\alpha \left(\log\frac{d_i}{\beta}\right)^2$$

$$\frac{\partial^2 L}{\partial\beta^2} = \frac{\alpha}{\beta^2}\left(N - (\alpha+1)\sum_i \left(\frac{d_i}{\beta}\right)^\alpha\right)$$

$$\frac{\partial^2 L}{\partial\alpha\partial\beta} = \frac{1}{\beta}\sum_i \left(\frac{d_i}{\beta}\right)^\alpha + \frac{\alpha}{\beta}\sum_i \left(\frac{d_i}{\beta}\right)^\alpha \log\frac{d_i}{\beta} - \frac{N}{\beta}$$

Note: there is a subtlety here which your implementation should address!

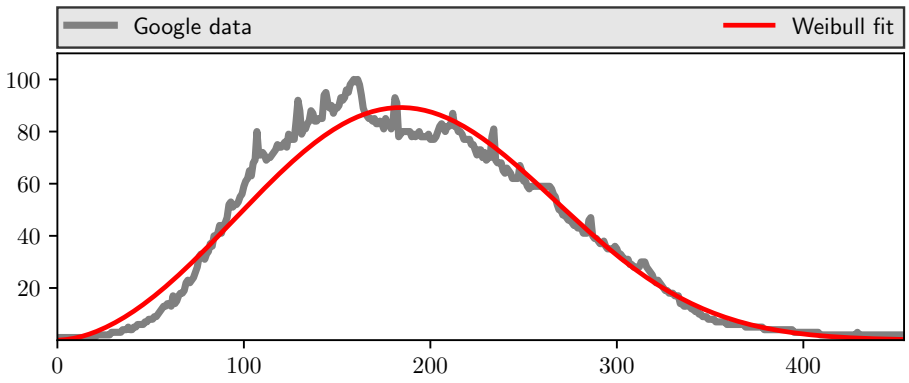
You are given a histogram $h[t]$ where h_j counts the number of searches at time t_j. The MLE procedure outlined above assumes that you are given individual observations d_i. That is, it assumes as input h_1 observations of a value t_1, h_2 observations observation of a value t_2, etc. In other words, $d_1 = t_1, d_2 = t_1, \ldots d_{h_1} = t_1, \ldots, d_{h_1+1} = t_2, \ldots, d_{h_1+h_2} = t_2, \ldots$

That is, it seems you would have to turn the histogram into a (large) set of numbers for the procedure to work. But there also is a more elegant solution, can see and implement it?

Be ambitious! Analyze why turning the histogram into a set of numbers is unnecessarily tedious and find and implement a faster approach.

If you initialize $\alpha = 1$ and $\beta = 1$ and run your estimation procedure for about 20 iterations, then which values do you obtain for α and β?

As a reference, when you plot the histogram and a correspondingly scaled version of your fitted Weibull distribution, your result should resemble this figure



Of course, this is a less than perfect explanation of the data we are dealing with but a much better one than a Gaussian could provide.

For those who are interested: There are further growth-and-decline models that would describe our data even better . . .

task 2.3 [10 points]

fitting a Weibull distribution to a histogram (part 2)

It is strange that neither `scipy.stats` nor `scipy.optimize` provide functionalities for fitting distributions to histograms, but we can hack our way around this . . .

Let

$$\boldsymbol{h} = \begin{bmatrix} h_1 & h_2 & \cdots & h_n \end{bmatrix}$$
$$\boldsymbol{t} = \begin{bmatrix} 1 & 2 & \cdots & n \end{bmatrix}$$

and $f(\,t \mid \alpha, \beta)$ be as in the previous task. Use the `scipy.optimize` function `curve_fit` to fit the following function to the histogram $h[t]$

$$\tilde{f}(\,t \mid A, \alpha, \beta) = A \cdot f(\,t \mid \alpha, \beta)$$

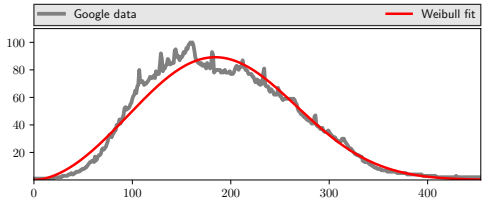
Observe that this is but a scaled version of the probability density function of the Weibull distribution where parameter A represents an amplitude.

Good initial values for the fitting procedure are

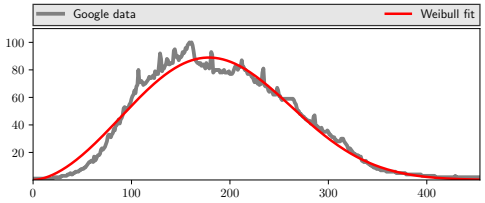
$$A = 1000$$
$$\alpha = 1$$
$$\beta = 1$$

If you use these, then which final estimates do you obtain for α and β?

Note: it would be surprising if your result was the same as in the previous task. For reference, here are plots of your instructors' results



typical result for task 2.2



typical result for task 2.3

task 2.4 [20 bonus points]

fitting a Weibull distribution to 1D data (part 3)

Note: This task is *not* mandatory but a bonus task. It gives motivated teams who want to widen their horizons an opportunity to earn extra points. The flip side is that it deals with more advanced ideas formulated in more abstract terms ...

Note: machine learning / mathematical model fitting is an art rather than a science! We have already seen two methods for fitting a Weibull distribution to histogram data. Both worked but led to slightly different results. Here is yet another method which will again yield slightly different results.

Above, you fitted a continuous distribution $f(t \mid \theta_1, \theta_2)$ to a discrete series of frequency counts $x_1, \ldots, x_n$ grouped into n distinct intervals $(t_0, t_1]$, $(t_1, t_2]$, ..., $(t_{n-1}, t_n]$.

To devise an efficient algorithm for estimating optimal model parameters $\hat{\theta}_1$ and $\hat{\theta}_2$, we note that a histogram of counts can also be thought of as a multinomial distribution

$$h(x_1, \ldots, x_n) = N! \prod_i \frac{p_i^{x_i}}{x_i!}$$

where

$$N = \sum_i x_i$$

Now, since the cumulative density of the model distribution $f(\, t \mid \theta_1, \theta_2)$ is

$$F(t) = F(t \mid \theta_1, \theta_2) = \int_0^t f(\tau \mid \theta_1, \theta_2)\, d\tau,$$

we may think of the probabilities p_i in the multinomial representation of our histogram as

$$p_i(\theta_1, \theta_2) = F(t_i) - F(t_{i-1})$$

so that

$$\sum_i p_i = F(t_n) - F(t_0)$$

Accordingly, the likelihood for a discrete (and possibly truncated) series of counts $x_1, \ldots, x_n$ is given by

$$L(\theta_1, \theta_2) = \frac{N!}{F(t_n) - F(t_0)} \prod_i \frac{p_i(\theta_1, \theta_2)^{x_i}}{x_i!}$$

and maximum likelihood estimates of θ_1 and θ_2 result from computing the roots of $\nabla_{\boldsymbol{\theta}} \log L$.

Again, this may not lead to closed form solutions but may require numerical optimization. To this end, we could apply an efficient, iterative weighted least squares scheme

$$\sum_i w_i \Big(x_i - N p_i(\theta_1, \theta_2) \Big)^2$$

which regresses the x_i onto their expectations $N p_i$ and requires to update the weights $w_i = (N p_i)^{-1}$ in each iteration.

In addition to computational convenience, this approach is robust and has the property that, for $p_i = p_i(\hat{\theta}_1, \hat{\theta}_2)$, the final residual sum of squares follows a χ^2 statistic which could be used for goodness-of-fit testing.

Finally, note that the cumulative density function of the Weibull distribution is given by

$$F(\, t \mid \alpha, \beta) = 1 - e^{-\left(\frac{t}{\beta}\right)^{\alpha}}$$

Given all this information, can you use the `scipy.optimize` function `leastsq` to implement the idea of multinomial maximum likelihood? If so, then

$$\alpha = 1$$
$$\beta = 100$$

are good initial guesses for fitting a Weibull distribution to the histogram data from the previous two tasks. If you use these, then which estimates do you obtain for α and β?

task 2.5 [10 points]

fitting a Weibull model to a discrete distribution

In the last couple of tasks, you fitted a continuous Weibull distribution

$$f(\,t \mid \alpha, \beta) = \tfrac{\alpha}{\beta}\big(\tfrac{t}{\beta}\big)^{\alpha-1}\, e^{-\left(\frac{t}{\beta}\right)^{\alpha}}$$

to a discrete histogram $\boldsymbol{h}$ with entries $h_j = h[t_j]$ counting search activities at certain points in time. Note that all methods considered so far implicitly discretized the Weibull distribution to $\boldsymbol{f} = [f_1, f_2, \ldots, f_n]$ where

$$f_j(\alpha, \beta) = \tfrac{\alpha}{\beta}\big(\tfrac{t_j}{\beta}\big)^{\alpha-1}\, e^{-\left(\frac{t_j}{\beta}\right)^{\alpha}}$$

Don't worry! All the methods are reasonable and, after all, any number in the memory of a digital computer is discrete anyway. But ...

Why not turn the histogram $\boldsymbol{h}$ into a discrete probability distribution $\boldsymbol{q}$ with entries $q_j = q[t_j]$ where

$$\begin{aligned} q_j &\geq 0 \\ \sum_j q_j &= 1 \end{aligned}$$

and fit the discrete distribution $\boldsymbol{f}$ to the discrete distribution $\boldsymbol{q}$ such that the Kullback-Leibler divergence

$$D_{KL}(\boldsymbol{f} \parallel \boldsymbol{q}) = \sum_j f_j \,\log \frac{f_j}{q_j}$$

between $\boldsymbol{f}$ and $\boldsymbol{q}$ becomes minimal?

Turning a histogram (of non-negative counts) into a distribution is easy, we usually simply have to normalize

$$q_j = \frac{h_j}{\sum_i h_i}$$

Note: however, in our particular setting, there is a catch. Our histogram $h[t_j]$ represents a times series or a collective attention process that evolves over time. Alas, due to the way our data was collected, we have no way of knowing whether or not the process has already ended (there still may be people today who are searching for *"myspace"*). We therefor have to make another modeling assumption ...

As our data suggest that interest in the search term *"myspace"* has more or less died out, we assume that the data we are given reflects 98% of the overall process.

Here is your task:

Use the following to turn the histogram $\boldsymbol{h}$ into an "open-ended" discrete probability distribution $\boldsymbol{q}$

$$q_j = \frac{h_j}{\sum_i h_i} \cdot 0.98$$

and then fit $\boldsymbol{f}$ with entries $f_j(\alpha, \beta)$ by minimizing $D_{KL}(\boldsymbol{f} \parallel \boldsymbol{q})$.

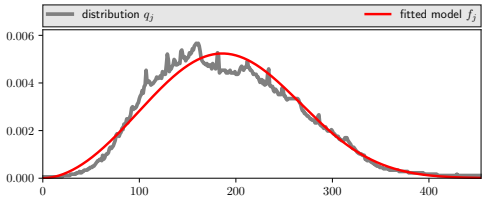
In and off itself, this is not a trivial task. However, if you figure out how to work with the `scipy.optimiz` function `minimize`, it should be rather straightforward.

Note: when working with `minimize`, please pay attention to the fact that the Weibull parameters α and β must be greater than 0, i.e. are lower bounded. Good initial guesses are

$$\alpha = 1$$
$$\beta = 100$$

If you use these, then which final estimates do you obtain for α and β?

Note: it would be once again surprising if your result was exactly the same as in the previous tasks. For reference, here is a plot of your instructors' results



typical result for task 2.5

task 2.6

submission of presentation and code

As always, prepare a presentation / set of slides on your solutions and results for tasks 1.2 and 1.3. These slides should help you to give a scientific presentation of your work (i.e. to give a short talk in front of your fellow students and instructors and answer any questions they may have).

W.r.t. to formalities, please make sure that

- your presentation contains a title slide which lists the names and matriculation numbers of everybody in your team who contributed to the solutions.

W.r.t. content, please make sure that

- your presentation contains about 12 to 15 content slides but not more
- your presentation is concise and clearly structured
- your presentation answers questions such as
  - "what was the task / problem we considered?"
  - "what kind of difficulties (if any) did we encounter?"
  - "how did we solve them?"
  - "what were our results?"
  - "what did we learn?"

Save / export your slides as a PDF file and upload it to eCampus.

As always, please name all your code files in a manner that indicates which task they solve and put them in an archive or a ZIP file.

Upload this archive / ZIP file to eCampus.