project 1:

warm up

solution(s) due:

Dec 3, 2015 at **12:00** via email to **bauckhag@bit.uni-bonn.de**

problem specification:

task 1.1: acquaint yourself with python for pattern recognition: first of all, download the file

`whExample.py`

from the Google site that accompanies the lecture; also retrieve the file

`whData.dat`

and run the exemplary script. It will plot sizes vs. weights of the students attending the lecture.

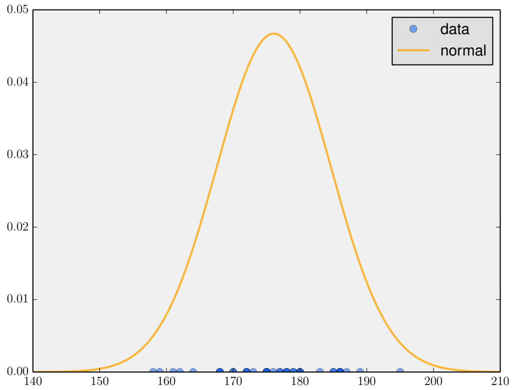
Note that two students did not want to disclose their weight. This provides us with an opportunity to deal with missing data. In the plot produced by the script, these data points appear with negative weights (they are *outliers*).

For starters, try to figure out a way of plotting the data without the outliers, that is, figure out how to plot only those data for which both measurements are positive.

task 1.2: fitting a Normal distribution to 1D data: now consider only the data on body sizes, i.e. consider the 1D array of data containing the size information.

Compute the mean and standard deviation of this sample, plot the data and a normal distribution characterizing its density.

Your result should resemble the following figure:



task 1.3: fitting a Weibull distribution to 1D data: download the file

`myspace.csv`

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

Read the data in the second column of the 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 $\boldsymbol{x} = [1, 2, \ldots, n]$.

Now, fit a Weibull distribution to the histogram $h(x)$. The probability density function of the Weibull distribution is given by

$$f(x \mid \kappa, \alpha) = \frac{\kappa}{\alpha} \left(\frac{x}{\alpha}\right)^{\kappa - 1} e^{-\left(\frac{x}{\alpha}\right)^{\kappa}}$$

where κ and α are a shape and scale parameter, respectively. Unlike in the case of the Normal distribution, maximum likelihood estimation of the parameters of the Weibull distribution is more involved.

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

$$L(\alpha, \kappa \mid D) = N\big(\log \kappa - \kappa \log \alpha\big) + (\kappa - 1) \sum_i \log d_i - \sum_i (d_i/\alpha)^{\kappa}.$$

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

$$\begin{bmatrix} \kappa^{\text{new}} \\ \alpha^{\text{new}} \end{bmatrix} = \begin{bmatrix} \kappa \\ \alpha \end{bmatrix} + \begin{bmatrix} \frac{\partial^2 L}{\partial \kappa^2} & \frac{\partial^2 L}{\partial \kappa \partial \alpha} \\ \frac{\partial^2 L}{\partial \kappa \partial \alpha} & \frac{\partial^2 L}{\partial \alpha^2} \end{bmatrix}^{-1} \begin{bmatrix} -\frac{\partial L}{\partial \kappa} \\ -\frac{\partial L}{\partial \alpha} \end{bmatrix}$$

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

$$\frac{\partial L}{\partial \kappa} = N/\kappa - N \log \alpha + \sum_i \log d_i - \sum_i (d_i/\alpha)^\kappa \log(d_i/\alpha)$$

$$\frac{\partial L}{\partial \alpha} = \kappa/\alpha \Big(\sum_i (d_i/\alpha)^\kappa - N \Big)$$

$$\frac{\partial^2 L}{\partial \kappa^2} = -N/\kappa^2 - \sum_i (d_i/\alpha)^\kappa \big(\log(d_i/\alpha) \big)^2$$

$$\frac{\partial^2 L}{\partial \alpha^2} = \kappa/\alpha^2 \Big(N - (\kappa + 1) \sum_i (d_i/\alpha)^\kappa \Big)$$

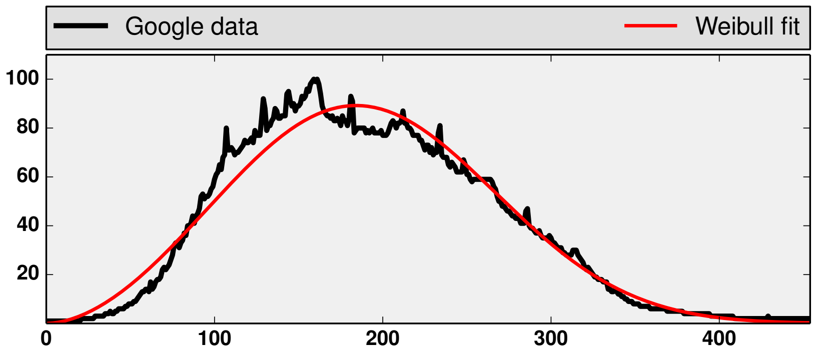
$$\frac{\partial^2 L}{\partial \kappa \partial \alpha} = 1/\alpha \sum_i (d_i/\alpha)^\kappa + \kappa/\alpha \sum_i (d_i/\alpha)^\kappa \log(d_i/\alpha) - N/\alpha.$$

note:

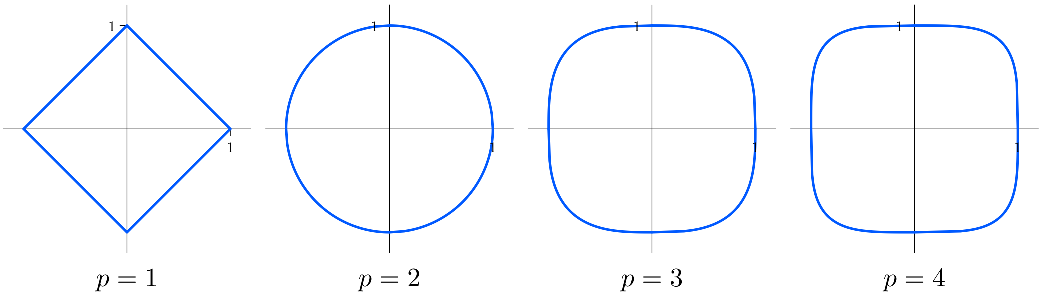
You are given a histogram $h(x)$ where h_j counts the number of observations of a value x_j. The MLE procedure outlined above assumes that you are given individual observations d_i. That is, it assumes as input h_1 times a value of x_1, h_2 times a value of x_2, etc. In other words, $d_1 = x_1, d_2 = x_1, \ldots d_{h_1} = x_1, \ldots d_{h_1+1} = x_2, \ldots, d_{h_1+h_2} = x_2, \ldots$ That is, you 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?

If you initialize $\kappa = 1$ and $\alpha = 1$ and run the estimation procedure for about 20 iterations, then which values do you obtain for κ and λ?

When you plot the histogram and a correspondingly scaled version of the fitted distribution, your result should resemble this figure:



task 1.4: drawing unit circles: in the lecture we discussed L_p norms for $\mathbb{R}^m$ and saw that, for different p, the corresponding unit spheres may look different. For instance, the following examples show unit circles in $\mathbb{R}^2$:

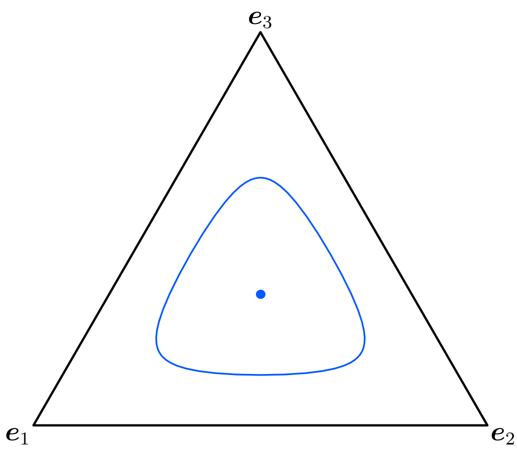


Consider the L_p norm for $p = \frac{1}{2}$ and plot the corresponding $\mathbb{R}^2$ unit circle.

bonus task: drawing circles in $\mathbb{S}^3$: completing this task is *not* mandatory, it is intended for those who consider the above problems trivial.

However, here is an incentive: if you can successfully complete this task, you can earn 5 additional points for the written exam at the end of the semester (which will be graded out of 80 points).

In the lecture, we discussed the Aitchison geometry of the vector space $\mathbb{S}^m$ and saw that, for $\mathbb{S}^3$ the corresponding unit circle (radius $r = 1$) looks like this:



Can you plot the circle centered at $\mathbf{0} \in \mathbb{S}^3$ that has a radius of $r = 3$?

general hints and remarks

- Send all your solutions (code, plots, slides) in a ZIP archive to

  bauckhag@bit.uni-bonn.de

- The goal of this project is to provide a gentle introduction to scientific python. There are numerous resources on the web related to python programming. Numpy and Scipy are more or less well documented and Matplotlib, too, comes with tons of tutorials. Play with the code that is provided. Most of the above tasks are trivial to solve, just look around for ideas as to how it can be done.
- **note:** if you insist on using a language other than python, you have to figure out elementary functions/toolboxes in these languages by yourself. **Implementations in MATLAB will not be accepted.**
- Remember that you have to complete all practical projects (and the tasks therein) to be eligible to the written exam at the end of the semester. Your grades (and credits) for this course will be decided based on the exam only, but –once again– you have to succeed in the projects to get there.
- Not handing in a solution implies failing the course.
- Your project work needs to be *satisfactory* to count as a success. Your code and results will be checked and your presentation needs to be convincing.
- If your solutions meets the above requirements and you can demonstrate that they work in practice, it is a *satisfactory* solution.
- A *good* to *very good* solution requires additional efforts especially w.r.t. to elegance and readability of your code. If your code is neither commented nor well structured, your solution is not good! A very good solution requires additional efforts towards the quality of your project presentation in the colloquium. Your presentation should be well timed, consistent, and convincing. Striving for very good solutions should always be your goal!