Probability and Statistics

Exercise sheet 11

Please ask questions in the exercise classes and/or post your questions (anonymously if you want) in this file: https://docs.google.com/document/d/1FuW9HQponei5ipS4j2J3lM4dfiP7MVQQ_0cMB1UUAXg/edit?usp=sharing

Exercise 11.1 Suppose that $X_1, \dots, X_n$ form a random sample from an absolutely continuous distribution for which the probability density function $f_\theta(x)$ under $\mathbb{P}_\theta$ is given by

$$f_\theta(x) = \begin{cases} \theta x^{\theta-1} & \text{for } 0 < x < 1, \\ 0 & \text{otherwise.} \end{cases}$$

Also, suppose that the value of $\theta > 0$ is unknown. Find the MLE (maximum likelihood estimator) for θ.

Exercise 11.2 Suppose that $X_1, \dots, X_n$ form a random sample from a normal distribution for which both the mean μ and the variance $\sigma^2 > 0$ are unknown.

(a) Find the MLE for $\theta = (\mu, \sigma^2)$.

(b) Find the MLE for $\tilde{\theta} = (\mu, \sigma)$.

Exercise 11.3 We toss a coin 100 times and we get 60 heads. We want to do a test to know whether the coin is fair. We can use the central limit theorem to approximate binomial distributions.

(a) Test the hypothesis with an approximate 0.01 level of significance. Should this test be one or two-sided (in other words, $\Theta_A = \{p : p > \frac{1}{2}\}$ or $\Theta_A = \{p : p \neq \frac{1}{2}\}$)? Compute the realized value of the P-value.

(b) What is the largest number of heads we should have in 100 tosses so that we cannot reject $\tilde{H}_0 :=$ "The coin is not biased towards head". Compute the realized value of the P-value.

Exercise 11.4 We want to study the `starsCYG`-dataset. Run the following **R**-code. (You can either install **R** and **R**-Studio or another **R**-environment or run it in your browser https://www.kaggle.com/jakobheiss/probstat2020-ex11-4/edit using a free kaggle-account or alternatively create a free RStudio Cloud account). (You could also use Anaconda to install R and RStudio.).

```
library(ggplot2);
library(reshape2);
library(robustbase);
library(MASS);
library(quantreg);


data(starsCYG);

data <- data.frame(log.Te = starsCYG$log.Te,
          LS = lm(log.light ~ ., data = starsCYG)$fitted.values,
          L1 = rq(log.light ~ ., data = starsCYG)$fitted.values,
          #M = rlm(log.light ~ ., data = starsCYG, method = "M")$fitted.values,
          S = lqs(log.light ~ ., data = starsCYG, method = "S")$fitted.values,
          MM = lmrob(log.light ~ ., data = starsCYG)$fitted.values,
        LMS = lqs(log.light ~ ., data = starsCYG, method = "lms")$fitted.values,
        LTS = ltsReg(log.light ~ ., data = starsCYG)$fitted.values
)

ggplot(melt(data, id.vars = "log.Te", variable.name = "Regression", value.name = "log.light"),
       aes(x = log.Te, y = log.light, color = Regression)) +
  geom_point(data = starsCYG, color = "#222222") +
  geom_line();
```

Each observation corresponds to a star. On the x-axis, the log of the temperature is plotted, and the y-axis corresponds to the brightness of the stars (log of the light-intensity). We want to find a (affine) linear regression line (see eq. (5.2.1) in the lecture notes) that helps us to understand the relationship between these two quantities. In the given code, there are 6 different methods applied to obtain a regression line. `LS` (least squares; see Example 5.2.3) and `L1` (L_1; see Example 6.2.3 in the lecture notes) have already been defined in the lecture. The other four methods `S`, `MM`, `LMS` and `LTS` are more advanced and you don't have to understand them in detail. These four algorithm all follow the goal to find a line that nicely fits through the majority of points instead of trying to fit all data points. The main idea of `LMS` is to minimize the median of the squared vertical distances between the line and the data points. Instead of the standard least squares method, which minimizes the sum of squared residuals over n points, the `LTS` (least trimmed squares) method attempts to minimize the sum of squared residuals over a subset consisting of h points. The unused $n - h$ points do not influence the fit. The main idea of `LTS` is quite well explained at `https://en.wikipedia.org/wiki/Least_trimmed_squares` (the **R**-implementation of `LTS` is a bit more advanced than that. If you want to test the basic version of LTS, you have to extract `raw.coefficients` from `LTS` and look at `h.alpha.n`). The focus of this exercise lies on having an interesting discussion in the exercise class and understanding intuitively the concept of outliers and breakpoints. You don't have to write down a perfect solution.

(a) Which of those stars are special (in other words, which follow a different pattern than the majority, or which are outliers)?

(b) In this example, one could solve (a) simply by looking at this two-dimensional plot. In high-dimensional situations, one cannot visualize the data so easily. Therefore, one needs algorithms that automatically detect (potential) outliers. One could have the idea to suspect those points with high vertical distance to the `LS`-regression-line to be the outliers. Do you think this is a good approach (in this example)? Do you have a better idea?

(c) Which of these algorithms have a breakpoint high enough for this problem?

If you have feedback regarding the exercise sheets, please send a mail to Jakob Heiss.